pageable-springbut/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"
—‡6“§66†VÖ Æö6 F–öãÒ&‡GG ¢òöÖ ven.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>sekolah
</groupId>
<artifactId>lms
</artifactId>
<version>0.0.1-SNAPSHOT
</version>
<name>lms
</name>
<description>Demo project for Spring Boot
</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>org.postgresql
</groupId>
<artifactId>postgresql
</artifactId>
<scope>runtime
</scope>
</dependency>
<dependency>
<groupId>org.projectlombok
</groupId>
<artifactId>lombok
</artifactId>
<optional>true
</optional>
</dependency>
<dependency>
<groupId>org.springframework.boot
</groupId>
<artifactId>spring-boot-starter-test
</artifactId>
<scope>test
</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot
</groupId>
<artifactId>spring-boot-starter-validation
</artifactId>
</dependency>
<!-- New dependencies for JWT and Security -->
<dependency>
<groupId>org.springframework.boot
</groupId>
<artifactId>spring-boot-starter-security
</artifactId>
</dependency>
<dependency>
<groupId>io.jsonwebtoken
</groupId>
<artifactId>jjwt-api
</artifactId>
<version>0.11.5
</version>
</dependency>
<dependency>
<groupId>io.jsonwebtoken
</groupId>
<artifactId>jjwt-impl
</artifactId>
<version>0.11.5
</version>
```

<scope>runtime
</scope>
</dependency>
<dependency>
<groupId>io.jsonwebtoken
</groupId>
<artifactId>jjwt-jackson
</artifactId>
<version>0.11.5
</version>
<scope>runtime
</scope>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot
</groupId>
<artifactId>spring-boot-maven-plugin
</artifactId>
<configuration>
<excludes>
<exclude>
<groupId>org.projectlombok
</groupId>
<artifactId>lombok
</artifactId>
</exclude>
</excludes>
</configuration>
</plugin>
</plugins>
</build>
</project>

pageable-springbut/postman-collection-rbac.json


```
{
  "info": {
    "_postman_id": "your-postman-id",
    "name": "LMS API (RBAC)",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "variable": [
    {
      "key": "base_url",
      "value": "http://localhost:8080"
    },
    {
      "key": "token",
      "value": ""
    }
  ],
  "item": [
    {
      "name": "Auth",
      "item": [
        {
          "name": "Sign Up (Student)",
          "request": {
            "method": "POST",
            "header": [],
            "body": {
              "mode": "raw",
              "raw": "{\n    \"username\": \"student1\",\n    \"password\": \"password123\",\n    \"role\":
\"ROLE_STUDENT\"\n}",
              "options": {
                "raw": {
                  "language": "json"
                }
              }
            },
            "url": {
              "raw": "{{base_url}}/auth/signup",
              "host": [
                "{{base_url}}"
              ],
              "path": [
                "auth",
                "signup"
              ]
            }
          },
          "response": []
        },
        {
          "name": "Sign Up (Teacher)",
          "request": {
            "method": "POST",
            "header": [],
            "body": {
```

```
            "mode": "raw",
            "raw": "{\n    \"username\": \"teacher1\",\n    \"password\": \"password123\",\n    \"role\":
\"ROLE_TEACHER\"\n}",
            "options": {
              "raw": {
                "language": "json"
              }
            }
          },
          "url": {
            "raw": "{{base_url}}/auth/signup",
            "host": [
              "{{base_url}}"
            ],
            "path": [
              "auth",
              "signup"
            ]
          }
        },
        "response": []
      },
      {
        "name": "Sign Up (Admin)",
        "request": {
          "method": "POST",
          "header": [],
          "body": {
            "mode": "raw",
            "raw": "{\n    \"username\": \"admin1\",\n    \"password\": \"password123\",\n    \"role\":
\"ROLE_ADMIN\"\n}",
            "options": {
              "raw": {
                "language": "json"
              }
            }
          },
          "url": {
            "raw": "{{base_url}}/auth/signup",
            "host": [
              "{{base_url}}"
            ],
            "path": [
              "auth",
              "signup"
            ]
          }
        },
        "response": []
      },
      {
        "name": "Sign In",
        "event": [
          {
            "listen": "test",
            "script": {
```

```
            "exec": [
              "var jsonData = pm.response.json();",
              "pm.collectionVariables.set(\"token\", jsonData.data.token);"
            ],
            "type": "text/javascript"
          }
        }
      ],
      "request": {
        "method": "POST",
        "header": [],
        "body": {
          "mode": "raw",
          "raw": "{\n    \"username\": \"student1\",\n    \"password\": \"password123\"\n}",
          "options": {
            "raw": {
              "language": "json"
            }
          }
        },
        "url": {
          "raw": "{{base_url}}/auth/signin",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "auth",
            "signin"
          ]
        }
      },
      "response": []
    }
  ]
},
{
  "name": "Students",
  "item": [
    {
      "name": "Create Student",
      "request": {
        "auth": {
          "type": "bearer",
          "bearer": [
            {
              "key": "token",
              "value": "{{token}}",
              "type": "string"
            }
          ]
        },
        "method": "POST",
        "header": [],
        "body": {
          "mode": "raw",
          "raw": "{\n    \"name\": \"John Doe\",\n    \"birthDate\": \"2000-01-01\"\n}",
```

```
      "options": {
        "raw": {
          "language": "json"
        }
      }
    },
    "url": {
      "raw": "{{base_url}}/students",
      "host": [
        "{{base_url}}"
      ],
      "path": [
        "students"
      ]
    }
  },
  "response": []
},
{
  "name": "Get All Students",
  "request": {
    "auth": {
      "type": "bearer",
      "bearer": [
        {
          "key": "token",
          "value": "{{token}}",
          "type": "string"
        }
      ]
    },
    "method": "GET",
    "header": [],
    "url": {
      "raw": "{{base_url}}/students?page=0&size=10&name=John",
      "host": [
        "{{base_url}}"
      ],
      "path": [
        "students"
      ],
      "query": [
        {
          "key": "page",
          "value": "0"
        },
        {
          "key": "size",
          "value": "10"
        },
        {
          "key": "name",
          "value": "John"
        }
      ]
    }
```

```
  },
  "response": []
},
{
  "name": "Get Student by ID",
  "request": {
    "auth": {
      "type": "bearer",
      "bearer": [
        {
          "key": "token",
          "value": "{{token}}",
          "type": "string"
        }
      ]
    },
    "method": "GET",
    "header": [],
    "url": {
      "raw": "{{base_url}}/students/1",
      "host": [
        "{{base_url}}"
      ],
      "path": [
        "students",
        "1"
      ]
    }
  },
  "response": []
},
{
  "name": "Update Student",
  "request": {
    "auth": {
      "type": "bearer",
      "bearer": [
        {
          "key": "token",
          "value": "{{token}}",
          "type": "string"
        }
      ]
    },
    "method": "PUT",
    "header": [],
    "body": {
      "mode": "raw",
      "raw": "{\n    \"name\": \"John Doe Updated\",\n    \"birthDate\": \"2000-01-01\"\n}",
      "options": {
        "raw": {
          "language": "json"
        }
      }
    },
    "url": {
```

```
          "raw": "{{base_url}}/students/1",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "students",
            "1"
          ]
        }
      },
      "response": []
    },
    {
      "name": "Delete Student",
      "request": {
        "auth": {
          "type": "bearer",
          "bearer": [
            {
              "key": "token",
              "value": "{{token}}",
              "type": "string"
            }
          ]
        },
        "method": "DELETE",
        "header": [],
        "url": {
          "raw": "{{base_url}}/students/1",
          "host": [
            "{{base_url}}"
          ],
          "path": [
            "students",
            "1"
          ]
        }
      },
      "response": []
    }
  ]
},
{
  "name": "Teachers",
  "item": [
    {
      "name": "Get All Teachers",
      "request": {
        "auth": {
          "type": "bearer",
          "bearer": [
            {
              "key": "token",
              "value": "{{token}}",
              "type": "string"
            }
```

```
            ]
          },
          "method": "GET",
          "header": [],
          "url": {
            "raw": "{{base_url}}/teachers",
            "host": [
              "{{base_url}}"
            ],
            "path": [
              "teachers"
            ]
          }
        },
        "response": []
      }
    ]
  },
  {
    "name": "Admin",
    "item": [
      {
        "name": "Get Admin Dashboard",
        "request": {
          "auth": {
            "type": "bearer",
            "bearer": [
              {
                "key": "token",
                "value": "{{token}}",
                "type": "string"
              }
            ]
          },
          "method": "GET",
          "header": [],
          "url": {
            "raw": "{{base_url}}/admin/dashboard",
            "host": [
              "{{base_url}}"
            ],
            "path": [
              "admin",
              "dashboard"
            ]
          }
        },
        "response": []
      }
    ]
  }
]
}
```

pageable-springbut/src/main/java/sekolah/lms/LmsApplication.java

```
package sekolah.lms;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LmsApplication {

— V lic static void main(String[] args) {
™SpringApplication.run(LmsApplication.class, args);
—Ð

}
```

pageable-springbut/src/main/java/sekolah/lms/config/SecurityConfig.java

```
package sekolah.lms.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import sekolah.lms.security.JwtAuthenticationFilter;

@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthFilter;

    public SecurityConfig(JwtAuthenticationFilter jwtAuthFilter) {
        this.jwtAuthFilter = jwtAuthFilter;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)  // New way to disable CSRF
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers("/auth/**").permitAll()
                        .requestMatchers(HttpMethod.GET, "/students/**").hasAnyRole("STUDENT",
"TEACHER", "ADMIN")
                        .requestMatchers(HttpMethod.POST, "/students/**").hasAnyRole("TEACHER", "ADMIN")
                        .requestMatchers(HttpMethod.PUT, "/students/**").hasAnyRole("TEACHER", "ADMIN")
                        .requestMatchers(HttpMethod.DELETE, "/students/**").hasRole("ADMIN")
                        .requestMatchers("/teachers/**").hasAnyRole("TEACHER", "ADMIN")
                        .requestMatchers("/admin/**").hasRole("ADMIN")
                        .anyRequest().authenticated()
                )
                .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
```

```
        return new BCryptPasswordEncoder();
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws
Exception {
        return config.getAuthenticationManager();
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/controller/AuthController.java

```
package sekolah.lms.controller;

import jakarta.validation.Valid;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;
import sekolah.lms.model.Role;
import sekolah.lms.model.User;
import sekolah.lms.repository.UserRepository;
import sekolah.lms.security.JwtTokenProvider;
import sekolah.lms.utils.Res;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/auth")
public class AuthController {

    private final AuthenticationManager authenticationManager;
    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
    private final JwtTokenProvider tokenProvider;

    public AuthController(AuthenticationManager authenticationManager, UserRepository userRepository,
                          PasswordEncoder passwordEncoder, JwtTokenProvider tokenProvider) {
        this.authenticationManager = authenticationManager;
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
        this.tokenProvider = tokenProvider;
    }

    @PostMapping("/signup")
    public ResponseEntity<?> registerUser(@Valid @RequestBody User user) {
        if (userRepository.findByUsername(user.getUsername()).isPresent()) {
            return ResponseEntity.badRequest().body("Username is already taken!");
        }

        user.setPassword(passwordEncoder.encode(user.getPassword()));

        // Default to ROLE_STUDENT if no role is provided
        if (user.getRole() == null) {
            user.setRole(Role.ROLE_STUDENT);
        }

        User result = userRepository.save(user);

        return ResponseEntity.ok("User registered successfully");
    }
```

```
    @PostMapping("/signin")
    public ResponseEntity<?> authenticateUser(@Valid @RequestBody User loginRequest) {
        Authentication authentication = authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(
                        loginRequest.getUsername(),
                        loginRequest.getPassword()
                )
        );

        SecurityContextHolder.getContext().setAuthentication(authentication);

        String jwt = tokenProvider.generateToken(authentication);
        Map<String, String> response = new HashMap<>();
        response.put("token", jwt);

        return Res.renderJson(response, "Login successful", org.springframework.http.HttpStatus.OK);
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/controller/ErrorController.java

```
package sekolah.lms.controller;

import jakarta.validation.ConstraintViolationException;
import jakarta.validation.UnexpectedTypeException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import sekolah.lms.utils.Res;

@RestControllerAdvice
public class ErrorController {
    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<?> handleConstraintViolationException(ConstraintViolationException e) {
        return Res.renderJson(null, e.getMessage(), HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler(UnexpectedTypeException.class)
    public ResponseEntity<?> handleUnexpectedType(UnexpectedTypeException e) {
        String message = e.getMessage();
        HttpStatus status = HttpStatus.BAD_REQUEST;

        if (e.getMessage().contains("birthDate")) {
            message = "birthDate cannot be blank";
        }
        return Res.renderJson(null,message, status);
    }

    @ExceptionHandler(RuntimeException.class)
    public ResponseEntity<?> handleRunTime(RuntimeException e) {
        String message = e.getMessage();
        HttpStatus status = HttpStatus.BAD_REQUEST;

        if (e.getMessage().contains("student with id")) {
            message = "student not found";
        }
        return Res.renderJson(null, message, HttpStatus.BAD_REQUEST);
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/controller/StudentController.java

```
package sekolah.lms.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.web.PageableDefault;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import sekolah.lms.model.Student;
import sekolah.lms.service.StudentService;
import sekolah.lms.utils.PageResponseWrapper;
import sekolah.lms.utils.Res;

import java.util.List;

@RestController
@RequestMapping("/students")
@RequiredArgsConstructor
public class StudentController {
    private final StudentService studentService;

    @PostMapping
    public Student create(@RequestBody Student request) {
        return studentService.create(request);
    }

    @GetMapping
    public ResponseEntity<?> getAll(
            @PageableDefault(size=10)Pageable pageable,
            @RequestParam(required = false) String name
            ) {
        return Res.renderJson(
                new PageResponseWrapper<>(studentService.getAll(pageable, name)),
                "KETEMU",
                HttpStatus.OK
        );
    }

    // car rent,
    // car, brand, users -> specfication, by name
    // car -> by name + by available

    @GetMapping("/{id}")
    public ResponseEntity<?> getOne(@PathVariable Integer id) {
        return Res.renderJson(
                studentService.getOne(id),
                "found",
                HttpStatus.OK
        );
    }

    // Validation
```

```
    // Specification -> getByName
    // JWT security
    // REST TEMPLATE -> backend konsumsi API dari luar
    // Docker ->

    //web response -> pagination
}
```

pageable-springbut/src/main/java/sekolah/lms/model/Role.java

```
package sekolah.lms.model;

public enum Role {
    ROLE_STUDENT,
    ROLE_TEACHER,
    ROLE_ADMIN
}
```

pageable-springbut/src/main/java/sekolah/lms/model/Student.java

```
package sekolah.lms.model;

import com.fasterxml.jackson.annotation.JsonFormat;
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import lombok.*;

import java.util.Date;

@Entity
@Table(name = "students")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotBlank
    private String name;
    private String nim;

    @NotNull
    @Column(name = "birth_date")
    @JsonFormat(pattern = "yyyy-MM-dd")
    @Temporal(TemporalType.DATE)
    private Date birthDate;
}
```

pageable-springbut/src/main/java/sekolah/lms/model/Subject.java

```
package sekolah.lms.model;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Table(name = "subjects")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Subject {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;

    @ManyToOne
    @JoinColumn(name="teacher_id", nullable = false)
    private Teacher teacher;
}
```

pageable-springbut/src/main/java/sekolah/lms/model/SubjectStudent.java

```
package sekolah.lms.model;

import jakarta.persistence.*;
import lombok.*;

import java.util.Date;
@Entity
@Table(name = "subject_students")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class SubjectStudent {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private Date startDate;
    private Date endDate;
    private Boolean completed;

    @ManyToOne
    @JoinColumn(name = "student_id")
    private Student student;

    @ManyToOne
    @JoinColumn(name = "subject_id")
    private Subject subject;
}
```

pageable-springbut/src/main/java/sekolah/lms/model/Teacher.java

```
package sekolah.lms.model;

import jakarta.persistence.*;
import lombok.*;

import java.util.Date;

@Entity
@Table(name = "teachers")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Teacher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
    @Column(name = "birth_date")
    private Date birthDate;
}
```

pageable-springbut/src/main/java/sekolah/lms/model/User.java

```
package sekolah.lms.model;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Table(name = "users")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Role role;
}
```

pageable-springbut/src/main/java/sekolah/lms/repository/StudentRepository.java

```
package sekolah.lms.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.stereotype.Repository;
import sekolah.lms.model.Student;

@Repository
public interface StudentRepository extends JpaRepository<Student, Integer>,
JpaSpecificationExecutor<Student> {
}
```

pageable-springbut/src/main/java/sekolah/lms/repository/UserRepository.java

```
package sekolah.lms.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import sekolah.lms.model.User;

import java.util.Optional;

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

pageable-springbut/src/main/java/sekolah/lms/security/CustomUserDetailsService.java

```
package sekolah.lms.security;

import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import sekolah.lms.model.User;
import sekolah.lms.repository.UserRepository;

import java.util.Collections;

@Service
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not found with username : " +
username));

        return new org.springframework.security.core.userdetails.User(user.getUsername(),
user.getPassword(),
                Collections.singletonList(new SimpleGrantedAuthority("ROLE_" + user.getRole())));
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/security/JwtAuthenticationFilter.java

```
package sekolah.lms.security;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtTokenProvider tokenProvider;
    private final CustomUserDetailsService customUserDetailsService;

    public JwtAuthenticationFilter(JwtTokenProvider tokenProvider, CustomUserDetailsService
customUserDetailsService) {
        this.tokenProvider = tokenProvider;
        this.customUserDetailsService = customUserDetailsService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
FilterChain filterChain)
            throws ServletException, IOException {
        try {
            String jwt = getJwtFromRequest(request);

            if (StringUtils.hasText(jwt) && tokenProvider.validateToken(jwt)) {
                String username = tokenProvider.getUsernameFromJWT(jwt);

                UserDetails userDetails = customUserDetailsService.loadUserByUsername(username);
                UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(
                        userDetails, null, userDetails.getAuthorities());
                authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(authentication);
            }
        } catch (Exception ex) {
            logger.error("Could not set user authentication in security context", ex);
        }

        filterChain.doFilter(request, response);
    }
```

```
    private String getJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");
        if (StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring(7);
        }
        return null;
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/security/JwtTokenProvider.java

```
package sekolah.lms.security;

import io.jsonwebtoken.*;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.stereotype.Component;

import java.security.Key;
import java.util.Date;
import java.util.stream.Collectors;

@Component
public class JwtTokenProvider {

    @Value("${app.jwtSecret}")
    private String jwtSecret;

    @Value("${app.jwtExpirationInMs}")
    private int jwtExpirationInMs;

    private final Key key;

    public JwtTokenProvider(@Value("${app.jwtSecret}") String jwtSecret) {
        this.key = Keys.hmacShaKeyFor(jwtSecret.getBytes());
    }

    public String generateToken(Authentication authentication) {
        String username = authentication.getName();
        String roles = authentication.getAuthorities().stream()
                .map(GrantedAuthority::getAuthority)
                .collect(Collectors.joining(","));

        Date now = new Date();
        Date expiryDate = new Date(now.getTime() + jwtExpirationInMs);

        return Jwts.builder()
                .setSubject(username)
                .claim("roles", roles)
                .setIssuedAt(new Date())
                .setExpiration(expiryDate)
                .signWith(key)
                .compact();
    }

    public String getUsernameFromJWT(String token) {
        Claims claims = Jwts.parserBuilder()
                .setSigningKey(key)
                .build()
                .parseClaimsJws(token)
                .getBody();
```

```
        return claims.getSubject();
    }

    public String getRolesFromJWT(String token) {
        Claims claims = Jwts.parserBuilder()
                .setSigningKey(key)
                .build()
                .parseClaimsJws(token)
                .getBody();

        return (String) claims.get("roles");
    }

    public boolean validateToken(String authToken) {
        try {
            Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(authToken);
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            // Log the exception
            return false;
        }
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/service/StudentService.java

```
package sekolah.lms.service;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import sekolah.lms.model.Student;

import java.util.List;


public interface StudentService {
    Student create(Student request);
    Page<Student> getAll(Pageable pageable, String name);
    Student getOne(Integer id);
    Student update(Integer id, Student request);
    void delete(Integer id);
}
```

pageable-springbut/src/main/java/sekolah/lms/service/impl/StudentServiceImpl.java

```
package sekolah.lms.service.impl;

import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;
import sekolah.lms.model.Student;
import sekolah.lms.repository.StudentRepository;
import sekolah.lms.service.StudentService;
import sekolah.lms.utils.specification.StudentSpecification;

import java.util.List;

@Service
@RequiredArgsConstructor
public class StudentServiceImpl implements StudentService {
    private final StudentRepository studentRepository;
    @Override
    public Student create(Student request) {
        String prefixNim = request.getName() + request.getBirthDate().getTime();
        request.setNim(prefixNim);
        return studentRepository.save(request);
    }

    @Override
    public Page<Student> getAll(Pageable pageable, String name) {
        Specification<Student> spec = StudentSpecification.getSpecification(name);
        return studentRepository.findAll(spec, pageable);
    }

    @Override
    public Student getOne(Integer id) {
        return studentRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("student with id: " + id + " not found"));
    }

    @Override
    public Student update(Integer id, Student request) {
        Student student = this.getOne(id);
        student.setName(request.getName());
        student.setNim(request.getNim());
        student.setBirthDate(request.getBirthDate());
        return studentRepository.save(student);
    }

    @Override
    public void delete(Integer id) {
        studentRepository.deleteById(id);
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/utils/PageResponseWrapper.java

```
package sekolah.lms.utils;

import lombok.Getter;
import lombok.Setter;
import org.springframework.data.domain.Page;

import java.util.List;

@Getter
@Setter
public class PageResponseWrapper<T>{
    private List<T> content;
    private Long totalElements;
    private Integer totalPages;
    private Integer page;
    private Integer size;

    public PageResponseWrapper(Page<T> page) {
        this.content = page.getContent();
        this.totalElements = page.getTotalElements();
        this.totalPages = page.getTotalPages();
        this.page = page.getNumber();
        this.size = page.getSize();
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/utils/Res.java

```
package sekolah.lms.utils;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

public class Res {
    public static <T> ResponseEntity<?> renderJson(T data, String message, HttpStatus httpStatus) {
        WebResponse<?> response = WebResponse.builder()
                .status(httpStatus.getReasonPhrase())
                .message(message)
                .data(data)
                .build();

        return ResponseEntity.status(httpStatus).body(response);
    }
}
```

pageable-springbut/src/main/java/sekolah/lms/utils/WebResponse.java

```
package sekolah.lms.utils;

import lombok.*;

@Getter
@Setter
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class WebResponse<T> {
    private String status;
    private String message;
    private T data;
}
```

pageable-springbut/src/main/java/sekolah/lms/utils/dto/StudentRequestDTO.java

```
package sekolah.lms.utils.dto;

import jakarta.validation.constraints.NotBlank;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class StudentRequestDTO {
    @NotBlank
    private String name;
}
```

pageable-springbut/src/main/java/sekolah/lms/utils/specification/StudentSpecification.java

```
package sekolah.lms.utils.specification;

import jakarta.persistence.criteria.Predicate;
import org.springframework.data.jpa.domain.Specification;
import sekolah.lms.model.Student;

import java.util.ArrayList;
import java.util.List;

public class StudentSpecification {
    public static Specification<Student> getSpecification(String name) {
        return (root, query, criteriaBuilder) -> {
            List<Predicate> predicates = new ArrayList<>();

            if (name != null && !name.isEmpty()) {
                predicates.add(criteriaBuilder.like(root.get("name"), "%" + name + "%"));
            }

            return criteriaBuilder.and(predicates.toArray(new Predicate[0]));
        };
    }
}
```

pageable-springbut/src/main/resources/application.properties

```
spring.application.name=lms

# Spring config
server.port=8080

# Database config
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/sekolah
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# JWT properties
app.jwtSecret=yourVeryLongAndSecureSecretKeyHerePleaseMakeItAtLeast256BitsLong
app.jwtExpirationInMs=86400000
```

pageable-springbut/src/test/java/sekolah/lms/LmsApplicationTests.java

```
package sekolah.lms;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class LmsApplicationTests {

” @est
—`oid contextLoads() {
—Ð

}
```

pageable-springbut/target/classes/application.properties

```
spring.application.name=lms

# Spring config
server.port=8080

# Database config
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/sekolah
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# JWT properties
app.jwtSecret=yourVeryLongAndSecureSecretKeyHerePleaseMakeItAtLeast256BitsLong
app.jwtExpirationInMs=86400000
```