BACKEND

Configuration

Package com.portfolio\_pro.app.configs;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class GlobalCorsConfig {

    @Bean

    WebMvcConfigurer corsConfigurer() {

        return new WebMvcConfigurer() {

            @Override

            public void addCorsMappings(CorsRegistry registry) {

                registry.addMapping("/\*\*")

                        .allowedOrigins("http://localhost:3000", "http://localhost:5173")

                        .allowedMethods("GET", "POST", "PUT", "DELETE", "PATCH")

                        .allowedHeaders("\*")

                        .allowCredentials(true);

            }

        };

    }

}

package com.portfolio\_pro.app.configs;

import java.io.IOException;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.utils.AuthUtil;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.Cookie;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired

    JwtService jwtService;

    @Autowired

    UserDetailsService userDetailsService;

    @Autowired

    AuthUtil authUtil;

    @Override

    protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, FilterChain filterChain)

            throws ServletException, IOException {

        Cookie[] cookies = httpServletRequest.getCookies();

        if(cookies == null) {

            filterChain.doFilter(httpServletRequest, httpServletResponse);

            return;

        }

        String authCookie = "";

        for(Cookie cookie: cookies) {

            if(cookie.getName().equals("AuthCookie")) {

                authCookie = cookie.getValue();

            }

        }

        if (authCookie.isEmpty()) {

            filterChain.doFilter(httpServletRequest, httpServletResponse);

            System.out.println("Lol");

            return;

        }

        final String jwt = authCookie;

        final String username = jwtService.extractUsername(jwt);

        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

        if (username != null && authentication == null) {

            UserDetails userDetails = userDetailsService.loadUserByUsername(username);

            System.out.println(userDetails.getUsername());

            if (jwtService.isTokenValid(jwt, userDetails)) {

                UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(

                        userDetails, null, userDetails.getAuthorities());

                authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(httpServletRequest));

                SecurityContextHolder.getContext().setAuthentication(authenticationToken);

            }

        }

        filterChain.doFilter(httpServletRequest, httpServletResponse);

    }

}

package com.portfolio\_pro.app.configs;

import java.util.List;

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

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.Customizer;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

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.core.userdetails.UserDetailsService;

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 org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.reactive.CorsConfigurationSource;

import org.springframework.web.cors.reactive.UrlBasedCorsConfigurationSource;

@Configuration

@EnableWebSecurity

public class WebSecurityConfig {

    @Autowired

    UserDetailsService userDetailsService;

    @Autowired

    JwtAuthenticationFilter jwtAuthenticationFilter;

    @Bean

    SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {

        httpSecurity.csrf(csrf -> csrf.disable()) // .cors(Customizer.withDefaults())

                .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

                .authorizeHttpRequests(request -> request.requestMatchers("/api/auth/signup", "/api/auth/signin")

                        .permitAll().requestMatchers("/api/public/\*\*", "/api/users/exists/\*\*").permitAll().anyRequest()

                        .authenticated())

                .httpBasic(Customizer.withDefaults())

                .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

        return httpSecurity.build();

    }

    /\*

    CorsConfigurationSource configurationSource() {

        CorsConfiguration corsConfiguration = new CorsConfiguration();

        corsConfiguration.setAllowedOrigins(List.of("http://localhost:3000","http://127.0.0.1:5173/", "http://vercel.app"));

        corsConfiguration.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "PATCH"));

        corsConfiguration.setAllowCredentials(true);

        corsConfiguration.setMaxAge(3600L);

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

        source.registerCorsConfiguration("/\*\*", corsConfiguration);

        return source;

    }

    \*/

    @Bean

    BCryptPasswordEncoder bCryptPasswordEncoder() {

        return new BCryptPasswordEncoder(14);

    }

    @Bean

    AuthenticationProvider authenticationProvider(UserDetailsService userDetailsService,

            PasswordEncoder passwordEncoder) {

        DaoAuthenticationProvider provider = new DaoAuthenticationProvider();

        provider.setUserDetailsService(userDetailsService);

        provider.setPasswordEncoder(passwordEncoder);

        return provider;

    }

    @Bean

    AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {

        return configuration.getAuthenticationManager();

    }

}

Controller

package com.portfolio\_pro.app.controllers;

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

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.ResponseStatus;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.services.AuthServices;

import jakarta.servlet.http.HttpServletResponse;

@RestController

@RequestMapping("/api/auth")

public class AuthController {

    @Autowired

    AuthServices authServices;

    @PostMapping("/signup")

    @ResponseStatus(HttpStatus.CREATED)

    public ResponseEntity<User> signup(@RequestBody UserSignupDto userSignupDto) throws UserException {

        return new ResponseEntity<User>(authServices.signupUser(userSignupDto), HttpStatus.CREATED);

    }

    @PostMapping("/signin")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<ResponseDto> signin(@RequestBody UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException{

        return new ResponseEntity<ResponseDto>(authServices.signinUser(userSigninDto, httpServletResponse), HttpStatus.OK);

    }

}

package com.portfolio\_pro.app.controllers;

import java.util.List;

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

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

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.ResponseStatus;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.services.PortfolioServices;

import jakarta.servlet.http.HttpServletRequest;

@RestController

@RequestMapping("/api/portfolios")

public class PortfolioController {

    @Autowired

    PortfolioServices portfolioServices;

    @PostMapping("/create-first")

    @ResponseStatus(HttpStatus.CREATED)

    public ResponseEntity<Portfolio> createFirstPortfolio(@RequestBody CreateFirstPortfolioDto createFirstPortfolioDto,

            HttpServletRequest httpServletRequest) throws UserException, PortfolioException {

        return new ResponseEntity<>(portfolioServices.createFirstPortfolio(createFirstPortfolioDto, httpServletRequest),

                HttpStatus.CREATED);

    }

    @GetMapping("/all")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<List<Portfolio>> getAllPortfoliosByUsername(HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        return new ResponseEntity<List<Portfolio>>(portfolioServices.getPortfoliosByUsername(httpServletRequest),

                HttpStatus.OK);

    }

    @GetMapping("/one/{portfolioId}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Portfolio> getPortfoliobyUserAndPortfolioId(HttpServletRequest httpServletRequest,

            @PathVariable Long portfolioId) throws UserException, PortfolioException {

        return new ResponseEntity<Portfolio>(

                portfolioServices.getPortfolioByUserAndPorfolioId(portfolioId, httpServletRequest), HttpStatus.OK);

    }

    @DeleteMapping("/remove/{portfolioId}")

    @ResponseStatus(HttpStatus.ACCEPTED)

    public ResponseEntity<ResponseDto> deletePortfolioByUserAndPortfolioId(HttpServletRequest httpServletRequest,

            @PathVariable Long portfolioId) throws UserException, PortfolioException {

        return new ResponseEntity<ResponseDto>(

                portfolioServices.deletePortfolioByUserAndPortfolioId(portfolioId, httpServletRequest),

                HttpStatus.ACCEPTED);

    }

}

package com.portfolio\_pro.app.controllers;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.services.PublicServices;

@RestController

@RequestMapping("/api/public")

public class PublicController {

    @Autowired

    PublicServices publicServices;

    @GetMapping("/{username}/{portfolioId}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Portfolio> getPortfolioByUsernameAndPortfolioId(@PathVariable String username, @PathVariable Long portfolioId) throws UserException, PortfolioException{

        return new ResponseEntity<Portfolio>(publicServices.getPortfolioByUsernameAndPortfolioId(username, portfolioId), HttpStatus.OK);

    }

}

package com.portfolio\_pro.app.controllers;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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.ResponseStatus;

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

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.services.UserServices;

import jakarta.servlet.http.HttpServletRequest;

@RestController

@RequestMapping("/api/users")

public class UserController {

    @Autowired

    UserServices userServices;

    @GetMapping("/exists/{username}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Boolean> isUsernameExists(@PathVariable String username) throws UserException {

        return new ResponseEntity<>(userServices.isUsernameExists(username), HttpStatus.OK);

    }

    @PostMapping("/update")

    @ResponseStatus(HttpStatus.ACCEPTED)

    public ResponseEntity<User> updateUser(@RequestBody UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException {

        return new ResponseEntity<>(userServices.updateUser(updateUserDto, httpServletRequest), HttpStatus.ACCEPTED);

    }

    @GetMapping("/me")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<User> getCurrentUser(HttpServletRequest httpServletRequest) throws UserException{

        return new ResponseEntity<User>(userServices.currentUser(httpServletRequest), HttpStatus.OK);

    }

}

DTO

package com.portfolio\_pro.app.dtos;

import java.util.List;

public class CreateFirstPortfolioDto {

//  portfolio data

    private String portfolioTitle;

    private String about;

    private String bio;

    private String professionalIdentity;

    private String profilePhotoUrl;

    private String skills;

    List<ExperienceDto> experienceDtos;

    List<ProjectDto> projectDtos;

    public CreateFirstPortfolioDto() {}

    public CreateFirstPortfolioDto(String portfolioTitle, String about, String bio, String professionalIdentity,

            String profilePhotoUrl, String skills, List<ExperienceDto> experienceDtos, List<ProjectDto> projectDtos) {

        this.portfolioTitle = portfolioTitle;

        this.about = about;

        this.bio = bio;

        this.professionalIdentity = professionalIdentity;

        this.profilePhotoUrl = profilePhotoUrl;

        this.experienceDtos = experienceDtos;

        this.projectDtos = projectDtos;

        this.skills = skills;

    }

    public String getPortfolioTitle() {

        return portfolioTitle;

    }

    public void setPortfolioTitle(String portfolioTitle) {

        this.portfolioTitle = portfolioTitle;

    }

    public String getAbout() {

        return about;

    }

    public void setAbout(String about) {

        this.about = about;

    }

    public String getBio() {

        return bio;

    }

    public void setBio(String bio) {

        this.bio = bio;

    }

    public String getProfessionalIdentity() {

        return professionalIdentity;

    }

    public void setProfessionalIdentity(String professionalIdentity) {

        this.professionalIdentity = professionalIdentity;

    }

    public String getProfilePhotoUrl() {

        return profilePhotoUrl;

    }

    public void setProfilePhotoUrl(String profilePhotoUrl) {

        this.profilePhotoUrl = profilePhotoUrl;

    }

    public List<ExperienceDto> getExperienceDtos() {

        return experienceDtos;

    }

    public void setExperienceDtos(List<ExperienceDto> experienceDtos) {

        this.experienceDtos = experienceDtos;

    }

    public List<ProjectDto> getProjectDtos() {

        return projectDtos;

    }

    public void setProjectDtos(List<ProjectDto> projectDtos) {

        this.projectDtos = projectDtos;

    }

    public String getSkills() {

        return skills;

    }

    public void setSkills(String skills) {

        this.skills = skills;

    }

}

package com.portfolio\_pro.app.dtos;

import java.time.LocalDateTime;

public class ErrorDetails {

    private String errorMessage;

    private String errorDetails;

    private LocalDateTime errorTimestamp;

    public ErrorDetails() {}

    public ErrorDetails(String errorMessage, String errorDetails, LocalDateTime errorTimestamp) {

        this.errorMessage = errorMessage;

        this.errorDetails = errorDetails;

        this.errorTimestamp = errorTimestamp;

    }

    public String getErrorMessage() {

        return errorMessage;

    }

    public void setErrorMessage(String errorMessage) {

        this.errorMessage = errorMessage;

    }

    public String getErrorDetails() {

        return errorDetails;

    }

    public void setErrorDetails(String errorDetails) {

        this.errorDetails = errorDetails;

    }

    public LocalDateTime getErrorTimestamp() {

        return errorTimestamp;

    }

    public void setErrorTimestamp(LocalDateTime errorTimestamp) {

        this.errorTimestamp = errorTimestamp;

    }

}

package com.portfolio\_pro.app.dtos;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonFormat;

public class ExperienceDto {

    private String jobTitle;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")

    private LocalDate startingDate;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")

    private LocalDate endingDate;

    private String description;

    private String company;

    public ExperienceDto() {}

    public ExperienceDto(String jobTitle, LocalDate startingDate, LocalDate endingDate, String description,

            String company) {

        this.jobTitle = jobTitle;

        this.startingDate = startingDate;

        this.endingDate = endingDate;

        this.description = description;

        this.company = company;

    }

    public String getJobTitle() {

        return jobTitle;

    }

    public void setJobTitle(String jobTitle) {

        this.jobTitle = jobTitle;

    }

    public LocalDate getStartingDate() {

        return startingDate;

    }

    public void setStartingDate(LocalDate startingDate) {

        this.startingDate = startingDate;

    }

    public LocalDate getEndingDate() {

        return endingDate;

    }

    public void setEndingDate(LocalDate endingDate) {

        this.endingDate = endingDate;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getCompany() {

        return company;

    }

    public void setCompany(String company) {

        this.company = company;

    }

}

package com.portfolio\_pro.app.dtos;

public class ProjectDto {

    private String projectName;

    private String description;

    private String link;

    private String technologies; // comma separated tech stack items

    public ProjectDto() {}

    public ProjectDto(String projectName, String description, String link, String technologies) {

        this.projectName = projectName;

        this.description = description;

        this.link = link;

        this.technologies = technologies;

    }

    public String getProjectName() {

        return projectName;

    }

    public void setProjectName(String projectName) {

        this.projectName = projectName;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getLink() {

        return link;

    }

    public void setLink(String link) {

        this.link = link;

    }

    public String getTechnologies() {

        return technologies;

    }

    public void setTechnologies(String technologies) {

        this.technologies = technologies;

    }

}

package com.portfolio\_pro.app.dtos;

public class ResponseDto {

    private String message;

    private Boolean success;

    public ResponseDto(String message, Boolean success) {

        this.message = message;

        this.success = success;

    }

    public ResponseDto() {}

    public String getMessage() {

        return message;

    }

    public void setMessage(String message) {

        this.message = message;

    }

    public Boolean getSuccess() {

        return success;

    }

    public void setSuccess(Boolean success) {

        this.success = success;

    }

}

package com.portfolio\_pro.app.dtos;

import java.util.List;

import com.portfolio\_pro.app.models.Education;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.SocialLink;

public class UpdateUserDto {

    private String username;

    private String email;

    private String password;

    private String fullName;

    private String phoneNo;

    private String location;

    private Portfolio activePortfolio;

    private List<Education> educations;

    private List<SocialLink> socialLinks;

    public UpdateUserDto() {

    }

    public UpdateUserDto(String username, String email, String password, String fullName, String phoneNo,

            String location, Portfolio activePortfolio, List<Education> educations, List<SocialLink> socialLinks) {

        this.username = username;

        this.email = email;

        this.password = password;

        this.fullName = fullName;

        this.phoneNo = phoneNo;

        this.location = location;

        this.activePortfolio = activePortfolio;

        this.educations = educations;

        this.socialLinks = socialLinks;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getFullName() {

        return fullName;

    }

    public void setFullName(String fullName) {

        this.fullName = fullName;

    }

    public String getPhoneNo() {

        return phoneNo;

    }

    public void setPhoneNo(String phoneNo) {

        this.phoneNo = phoneNo;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String location) {

        this.location = location;

    }

    public List<Education> getEducations() {

        return educations;

    }

    public void setEducations(List<Education> educations) {

        this.educations = educations;

    }

    public List<SocialLink> getSocialLinks() {

        return socialLinks;

    }

    public void setSocialLinks(List<SocialLink> socialLinks) {

        this.socialLinks = socialLinks;

    }

    public Portfolio getActivePortfolio() {

        return activePortfolio;

    }

    public void setActivePortfolio(Portfolio activePortfolio) {

        this.activePortfolio = activePortfolio;

    }

}

Models

package com.portfolio\_pro.app.models;

import java.util.Collection;

import java.util.Collections;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetails implements UserDetails{

    private static final long serialVersionUID = 1L;

    private final User user;

    public CustomUserDetails(User user) {

        this.user = user;

    }

    @Override

    public Collection<? extends GrantedAuthority> getAuthorities() {

        return Collections.singleton(new SimpleGrantedAuthority("USER"));

    }

    @Override

    public String getPassword() {

        return user.getPassword();

    }

    @Override

    public String getUsername() {

        return user.getUsername();

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

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 jakarta.persistence.Table;

@Entity

@Table(name = "educations")

public class Education {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "education\_id")

    private Long educationId;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @Column(name = "degree\_name")

    private String degreeName;

    @Column(name = "institute\_name")

    private String instituteName;

    @Column(name = "starting\_year")

    private Integer startingYear;

    @Column(name = "ending\_year")

    private Integer endingYear;

    @Column(columnDefinition = "TEXT", name = "description")

    private String description;

    public Education() {

    }

    public Long getEducationId() {

        return educationId;

    }

    public void setEducationId(Long educationId) {

        this.educationId = educationId;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public String getDegreeName() {

        return degreeName;

    }

    public void setDegreeName(String degreeName) {

        this.degreeName = degreeName;

    }

    public String getInstituteName() {

        return instituteName;

    }

    public void setInstituteName(String instituteName) {

        this.instituteName = instituteName;

    }

    public Integer getStartingYear() {

        return startingYear;

    }

    public void setStartingYear(Integer startingYear) {

        this.startingYear = startingYear;

    }

    public Integer getEndingYear() {

        return endingYear;

    }

    public void setEndingYear(Integer endingYear) {

        this.endingYear = endingYear;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

}

package com.portfolio\_pro.app.models;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Column;

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 jakarta.persistence.Table;

@Entity

@Table(name = "experiences")

public class Experience {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "experience\_id")

    private Long experienceId;

    @ManyToOne

    @JoinColumn(name = "portfolio\_id", nullable = false)

    @JsonIgnore

    private Portfolio portfolio;

    @Column(name = "job\_title")

    private String jobTitle;

    @Column(name = "company")

    private String company;

    @Column(name = "starting\_date")

    private LocalDate startingDate;

    @Column(name = "ending\_date")

    private LocalDate endingDate;

    @Column(columnDefinition = "TEXT", name = "description")

    private String description;

    public Experience() {}

    public Long getExperienceId() {

        return experienceId;

    }

    public void setExperienceId(Long experienceId) {

        this.experienceId = experienceId;

    }

    public Portfolio getPortfolio() {

        return portfolio;

    }

    public void setPortfolio(Portfolio portfolio) {

        this.portfolio = portfolio;

    }

    public String getJobTitle() {

        return jobTitle;

    }

    public void setJobTitle(String jobTitle) {

        this.jobTitle = jobTitle;

    }

    public String getCompany() {

        return company;

    }

    public void setCompany(String company) {

        this.company = company;

    }

    public LocalDate getStartingDate() {

        return startingDate;

    }

    public void setStartingDate(LocalDate startingDate) {

        this.startingDate = startingDate;

    }

    public LocalDate getEndingDate() {

        return endingDate;

    }

    public void setEndingDate(LocalDate endingDate) {

        this.endingDate = endingDate;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

}

package com.portfolio\_pro.app.models;

import java.util.List;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GenerationType;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

@Entity

@Table(name = "portfolios")

public class Portfolio {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "portfolio\_id")

    private long portfolioId;

    @Column(name = "portfolio\_title")

    private String portfolioTitle;

    @Column(name = "bio", columnDefinition = "TEXT")

    private String bio;

    @Column(name = "about", columnDefinition = "LONGTEXT")

    private String about;

    @Column(name = "professional\_identity")

    private String professionalIdentity;

    @Column(name = "profile\_photo\_url", unique = true)

    private String profilePhotoUrl;

    @Column(name = "skills", columnDefinition = "TEXT")

    private String skills;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @OneToMany(mappedBy = "portfolio", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Project> projects;

    @OneToMany(mappedBy = "portfolio", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Experience> experiences;

    public Portfolio() {

    }

    public long getPortfolioId() {

        return portfolioId;

    }

    public void setPortfolioId(long portfolioId) {

        this.portfolioId = portfolioId;

    }

    public String getPortfolioTitle() {

        return portfolioTitle;

    }

    public void setPortfolioTitle(String portfolioTitle) {

        this.portfolioTitle = portfolioTitle;

    }

    public String getBio() {

        return bio;

    }

    public void setBio(String bio) {

        this.bio = bio;

    }

    public String getAbout() {

        return about;

    }

    public void setAbout(String about) {

        this.about = about;

    }

    public String getProfessionalIdentity() {

        return professionalIdentity;

    }

    public void setProfessionalIdentity(String professionalIdentity) {

        this.professionalIdentity = professionalIdentity;

    }

    public String getProfilePhotoUrl() {

        return profilePhotoUrl;

    }

    public void setProfilePhotoUrl(String profilePhotoUrl) {

        this.profilePhotoUrl = profilePhotoUrl;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public List<Project> getProjects() {

        return projects;

    }

    public void setProjects(List<Project> projects) {

        this.projects = projects;

    }

    public List<Experience> getExperiences() {

        return experiences;

    }

    public void setExperiences(List<Experience> experiences) {

        this.experiences = experiences;

    }

    public String getSkills() {

        return skills;

    }

    public void setSkills(String skills) {

        this.skills = skills;

    }

}

package com.portfolio\_pro.app.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Column;

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 jakarta.persistence.Table;

@Entity

@Table(name = "projects")

public class Project {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long projectId;

    @ManyToOne

    @JoinColumn(name = "portfolio\_id", nullable = false)

    @JsonIgnore

    private Portfolio portfolio;

    @Column(columnDefinition = "TEXT", name = "project\_name")

    private String projectName;

    @Column(columnDefinition = "LONGTEXT", name = "description")

    private String description;

    @Column(unique = true, name = "link")

    private String link;

    @Column(columnDefinition = "TEXT", name = "technologies")

    private String technologies;

    public Project() {}

    public Long getProjectId() {

        return projectId;

    }

    public void setProjectId(Long projectId) {

        this.projectId = projectId;

    }

    public Portfolio getPortfolio() {

        return portfolio;

    }

    public void setPortfolio(Portfolio portfolio) {

        this.portfolio = portfolio;

    }

    public String getProjectName() {

        return projectName;

    }

    public void setProjectName(String projectName) {

        this.projectName = projectName;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getLink() {

        return link;

    }

    public void setLink(String link) {

        this.link = link;

    }

    public String getTechnologies() {

        return technologies;

    }

    public void setTechnologies(String technologies) {

        this.technologies = technologies;

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

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 jakarta.persistence.Table;

@Entity

@Table(name = "social\_links")

public class SocialLink {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "social\_id")

    private Long socialId;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @Column(name = "platform\_name")

    private String platformName;

    @Column(unique = true, name = "social\_url")

    private String socialUrl;

    public SocialLink() {}

    public Long getSocialId() {

        return socialId;

    }

    public void setSocialId(Long socialId) {

        this.socialId = socialId;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public String getPlatformName() {

        return platformName;

    }

    public void setPlatformName(String platformName) {

        this.platformName = platformName;

    }

    public String getSocialUrl() {

        return socialUrl;

    }

    public void setSocialUrl(String socialUrl) {

        this.socialUrl = socialUrl;

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.OneToMany;

import jakarta.persistence.OneToOne;

import jakarta.persistence.Table;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.CascadeType;

@Entity

@Table(name = "users")

public class User {

    @Id

    @Column(name = "username")

    private String username;

    @Column(name = "email", unique = true, nullable = false)

    private String email;

    @Column(name = "password", nullable = false)

    private String password;

    @Column(name = "full\_name", nullable = false)

    private String fullName;

    @Column(name = "phone\_no", unique = true)

    private String phoneNo;

    @Column(name = "location")

    private String location;

    @OneToOne

    @JoinColumn(name = "active\_portfolio\_id")

    @JsonIgnore

    private Portfolio activePortfolio;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    @JsonIgnore

    private List<Portfolio> portfolios;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Education> educations;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<SocialLink> socialLinks;

    public User() {

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getFullName() {

        return fullName;

    }

    public void setFullName(String fullName) {

        this.fullName = fullName;

    }

    public String getPhoneNo() {

        return phoneNo;

    }

    public void setPhoneNo(String phoneNo) {

        this.phoneNo = phoneNo;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String location) {

        this.location = location;

    }

    public Portfolio getActivePortfolio() {

        return activePortfolio;

    }

    public void setActivePortfolio(Portfolio activePortfolio) {

        this.activePortfolio = activePortfolio;

    }

    public List<Portfolio> getPortfolios() {

        return portfolios;

    }

    public void setPortfolios(List<Portfolio> portfolios) {

        this.portfolios = portfolios;

    }

    public List<Education> getEducations() {

        return educations;

    }

    public void setEducations(List<Education> educations) {

        this.educations = educations;

    }

    public List<SocialLink> getSocialLinks() {

        return socialLinks;

    }

    public void setSocialLinks(List<SocialLink> socialLinks) {

        this.socialLinks = socialLinks;

    }

}

Repositories

package com.portfolio\_pro.app.repositories;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Education;

@Repository

public interface EducationRepository extends JpaRepository<Education, Long>{

}

package com.portfolio\_pro.app.repositories;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Experience;

@Repository

public interface ExperienceRepositry extends JpaRepository<Experience, Long>{

    @SuppressWarnings("unchecked")

    public Experience save(Experience experience);

}

package com.portfolio\_pro.app.repositories;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.User;

@Repository

public interface PortfolioRepository extends JpaRepository<Portfolio, Long>{

    @SuppressWarnings("unchecked")

    public Portfolio save(Portfolio portfolio);

    public List<Portfolio> findByUser(User user);

    public Optional<Portfolio> findByUserAndPortfolioId(User user, Long portfolioId);

}

package com.portfolio\_pro.app.repositories;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Project;

@Repository

public interface ProjectRepository extends JpaRepository<Project, Long>{

    @SuppressWarnings("unchecked")

    public Project save(Project project);

}

package com.portfolio\_pro.app.repositories;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.SocialLink;

@Repository

public interface SocialLinkRepository extends JpaRepository<SocialLink, Long>{

}

package com.portfolio\_pro.app.repositories;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.User;

@Repository

public interface UserRepository extends JpaRepository<User, String>{

    public boolean existsByUsername(String username);

    public boolean existsByEmail(String email);

    public Optional<User> findByUsername(String username);

    @SuppressWarnings("unchecked")

    public User save(User user);

}

Serviceimplementation

package com.portfolio\_pro.app.serviceimps;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.AuthServices;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.http.Cookie;

import jakarta.servlet.http.HttpServletResponse;

@Service

public class AuthServicesImp implements AuthServices {

    @Autowired

    BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired

    UserRepository userRepository;

    @Autowired

    AuthenticationManager authenticationManager;

    @Autowired

    JwtService jwtService;

    @Autowired

    AuthUtil authUtil;

    @Override

    public User signupUser(UserSignupDto userSignupDto) throws UserException{

        if(userRepository.existsByEmail(userSignupDto.getEmail())) {

            throw new UserException("Email is already in use");

        }

        if(userRepository.existsByUsername(userSignupDto.getUsername())) {

            throw new UserException("Username is already taken");

        }

        User user = ModelMapper.toUserFromUserSignupDto(userSignupDto);

        user.setPassword(bCryptPasswordEncoder.encode(userSignupDto.getPassword()));

        return userRepository.save(user);

    }

    @Override

    public ResponseDto signinUser(UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException{

        Authentication authentication = authenticationManager

                .authenticate(new UsernamePasswordAuthenticationToken(userSigninDto.getUsername(), userSigninDto.getPassword()));

        if(authentication.isAuthenticated()) {

            Cookie cookie = authUtil.generateCookie(userSigninDto);

            httpServletResponse.addCookie(cookie);

            return new ResponseDto(authUtil.generateCookie(userSigninDto).getValue(), true);

        }

        return new ResponseDto(null, false);

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ExperienceDto;

import com.portfolio\_pro.app.dtos.ProjectDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Experience;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.Project;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.ExperienceRepositry;

import com.portfolio\_pro.app.repositories.PortfolioRepository;

import com.portfolio\_pro.app.repositories.ProjectRepository;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.services.PortfolioServices;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.transaction.Transactional;

@Service

public class PortfolioServicesImp implements PortfolioServices {

    @Autowired

    ProjectRepository projectRepository;

    @Autowired

    UserRepository userRepository;

    @Autowired

    PortfolioRepository portfolioRepository;

    @Autowired

    ExperienceRepositry experienceRepositry;

    @Autowired

    JwtService jwtService;

    @Autowired

    AuthUtil authUtil;

    @Override

    @Transactional

    public Portfolio createFirstPortfolio(CreateFirstPortfolioDto createFirstPortfolioDto,

            HttpServletRequest httpServletRequest) throws PortfolioException, UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User not available by username " + username);

        }

        Portfolio portfolio = ModelMapper.toPortfolioFromCreateFirstPortfolioDto(createFirstPortfolioDto);

        portfolio.setUser(user.get());

        Portfolio newPortfolio = portfolioRepository.save(portfolio);

        User currUser = user.get();

        currUser.setActivePortfolio(newPortfolio);

        User updatedUser = userRepository.save(currUser);

        if (updatedUser == null) {

            throw new UserException("Failed to update active portfolio.");

        }

        List<ExperienceDto> experienceDtos = createFirstPortfolioDto.getExperienceDtos();

        List<ProjectDto> projectDtos = createFirstPortfolioDto.getProjectDtos();

        for (ExperienceDto experienceDto : experienceDtos) {

            Experience experience = ModelMapper.toExperienceFromExperienceDto(experienceDto);

            experience.setPortfolio(newPortfolio);

            experienceRepositry.save(experience);

        }

        for (ProjectDto projectDto : projectDtos) {

            Project project = ModelMapper.toProjectFromProjectDto(projectDto);

            project.setPortfolio(newPortfolio);

            projectRepository.save(project);

        }

        newPortfolio = portfolioRepository.save(newPortfolio);

        if (newPortfolio == null) {

            throw new PortfolioException("Failed to Create Portfolio");

        }

        return newPortfolio;

    }

    @Override

    public List<Portfolio> getPortfoliosByUsername(HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        List<Portfolio> portfolios = portfolioRepository.findByUser(user.get());

        return portfolios;

    }

    @Override

    public Portfolio getPortfolioByUserAndPorfolioId(Long portfolioId, HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findByUserAndPortfolioId(user.get(), portfolioId);

        if (portfolio.isEmpty()) {

            throw new PortfolioException("Portfolio Does Not Exist");

        }

        return portfolio.get();

    }

    @Override

    @Transactional

    public ResponseDto deletePortfolioByUserAndPortfolioId(Long portfolioId, HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findById(portfolioId);

        if (portfolio.isEmpty()) {

            throw new PortfolioException("Portfolio Does Not Exists");

        }

        if (!portfolio.get().getUser().getUsername().equals(username)) {

            throw new UserException("User Not Authorized");

        }

        if (user.get().getActivePortfolio() != null

                && user.get().getActivePortfolio().getPortfolioId() == portfolio.get().getPortfolioId()) {

            user.get().setActivePortfolio(null);

            userRepository.save(user.get()); // update user to remove reference

        }

        portfolioRepository.delete(portfolio.get());

        return new ResponseDto("Portfolio Successfully Deleted", true);

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.PortfolioRepository;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.PublicServices;

@Service

public class PublicServiceImp implements PublicServices{

    @Autowired

    UserRepository userRepository;

    @Autowired

    PortfolioRepository portfolioRepository;

    @Override

    public Portfolio getPortfolioByUsernameAndPortfolioId(String username, Long portfolioId)

            throws UserException, PortfolioException {

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findByUserAndPortfolioId(user.get(), portfolioId);

        if(portfolio.isEmpty()) {

            throw new UserException("Portfolio Does Not Exists");

        }

        return portfolio.get();

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.UserServices;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

@Service

public class UserServicesImp implements UserServices{

    @Autowired

    UserRepository userRepository;

    @Autowired

    AuthUtil authUtil;

    @Override

    public boolean isUsernameExists(String username) {

        return userRepository.existsByUsername(username);

    }

    @Override

    public User updateUser(UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        User updatedUserData = ModelMapper.mapUserDtoToUser(user.get(), updateUserDto);

        User updatedUser = userRepository.save(updatedUserData);

        if(updatedUser == null) {

            throw new UserException("User Update Failed");

        }

        return updatedUser;

    }

    @Override

    public User currentUser(HttpServletRequest httpServletRequest) throws UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        return user.get();

    }

}

Service

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import jakarta.servlet.http.HttpServletResponse;

public interface AuthServices {

    public User signupUser(UserSignupDto userSignupDto) throws UserException;

    public ResponseDto signinUser(UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException;

}

package com.portfolio\_pro.app.services;

import java.util.Optional;

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

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 com.portfolio\_pro.app.models.CustomUserDetails;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

@Service

public class CustomUserDetailsService implements UserDetailsService{

    @Autowired

    UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {

        this.userRepository = userRepository;

    }

    @Override

    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            System.out.println("User not available");

            throw new UsernameNotFoundException("User not found!!!");

        }

        return new CustomUserDetails(user.get());

    }

}

package com.portfolio\_pro.app.services;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

import javax.crypto.SecretKey;

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

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.io.Decoders;

import io.jsonwebtoken.security.Keys;

@Service

public class JwtService {

    @Value("${jwt.secret}")

    private String secretKey;

    public String generateToken(UserSigninDto userSigninDto) {

        Map<String, Object> claims = new HashMap<>();

        claims.put("role", "USER");

        return Jwts

                .builder()

                .claims()

                .add(claims)

                .subject(userSigninDto.getUsername())

                .issuer("XERK").issuedAt(new Date(System.currentTimeMillis()))

                .expiration(new Date(System.currentTimeMillis() + 24\*60\*60\*1000))

                .and()

                .signWith(generateKey())

                .compact();

    }

    private SecretKey generateKey() {

        byte[] decode = Decoders.BASE64.decode(getSecretKey());

        return Keys.hmacShaKeyFor(decode);

    }

    public String getSecretKey() {

        return secretKey;

    }

    public boolean isTokenValid(String jwt, UserDetails userDetails) {

        final String username = extractUsername(jwt);

        return (userDetails.getUsername().equals(username) && !isTokenExpired(jwt));

    }

    private boolean isTokenExpired(String jwt) {

        return extractExpiration(jwt).before(new Date());

    }

    private Date extractExpiration(String jwt) {

        return extractClaims(jwt, Claims::getExpiration);

    }

    public String extractUsername(String jwt) {

        return extractClaims(jwt, Claims::getSubject);

    }

    private <T> T extractClaims(String jwt, Function<Claims, T> claimResolver) {

        Claims claims = extractClaims(jwt);

        return claimResolver.apply(claims);

    }

    private Claims extractClaims(String jwt) {

        return Jwts.parser().verifyWith(generateKey()).build().parseSignedClaims(jwt).getPayload();

    }

}

package com.portfolio\_pro.app.services;

import java.util.List;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import jakarta.servlet.http.HttpServletRequest;

public interface PortfolioServices {

    public Portfolio createFirstPortfolio(CreateFirstPortfolioDto createPortfolioDto, HttpServletRequest httpServletRequest) throws PortfolioException, UserException;

    public List<Portfolio> getPortfoliosByUsername(HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

    public Portfolio getPortfolioByUserAndPorfolioId(Long portfolioId, HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

    public ResponseDto deletePortfolioByUserAndPortfolioId(Long portfolioId, HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

}

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

public interface PublicServices {

    public Portfolio getPortfolioByUsernameAndPortfolioId(String username, Long portfolioId) throws UserException, PortfolioException;

}

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import jakarta.servlet.http.HttpServletRequest;

public interface UserServices {

     public boolean isUsernameExists(String username);

//   public boolean changeUsername(String username) throws UserException;

     public User updateUser(UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException;

     public User currentUser(HttpServletRequest httpServletRequest) throws UserException;

}

package com.portfolio\_pro.app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class PortfolioProApplication {

    public static void main(String[] args) {

        SpringApplication.run(PortfolioProApplication.class, args);

    }

}