themes-one D:\project\JAVA PROJECT\themes-one

=

src

=

CONFIG

© AppConfig : package com.themes.config;  
  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
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.provisioning.InMemoryUserDetailsManager;  
  
  
@Configuration  
public class AppConfig {  
  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration builder) throws Exception {  
 return builder.getAuthenticationManager();  
 }  
   
   
}

SecurityConfig: package com.themes.config;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.http.SessionCreationPolicy;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
  
import com.themes.security.JWTAuthenticationEntryPoint;  
import com.themes.security.JWTAuthenticationFilter;  
  
@Configuration  
public class SecurityConfig {  
   
 @Autowired  
 private JWTAuthenticationEntryPoint point;  
 @Autowired  
 private JWTAuthenticationFilter filter;  
   
 @Autowired  
 private UserDetailsService userDetailsService;  
   
 @Autowired  
 private PasswordEncoder passwordEncoder;  
   
   
   
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
  
 http.csrf(csrf -> csrf.disable())  
 .cors(cors->cors.disable())  
 .authorizeHttpRequests(  
 auth-> auth.requestMatchers("/api/\*\*")  
 .authenticated().requestMatchers("auth/create-user").permitAll()  
 .requestMatchers("auth/login").permitAll()  
 .requestMatchers("/reset-password/reset").permitAll()  
 .anyRequest().authenticated())  
 .exceptionHandling(ex-> ex.authenticationEntryPoint(point))  
 .sessionManagement(session->session.sessionCreationPolicy(SessionCreationPolicy.STATELESS));  
   
 http.addFilterBefore(filter, UsernamePasswordAuthenticationFilter.class);  
   
 return http.build();  
 }  
   
   
 @Bean  
 public DaoAuthenticationProvider doDaoAuthenticationProvider()  
 {  
 DaoAuthenticationProvider daoAuthenticationProvider = new DaoAuthenticationProvider();  
 daoAuthenticationProvider.setUserDetailsService(userDetailsService);  
 daoAuthenticationProvider.setPasswordEncoder(passwordEncoder);  
   
 return daoAuthenticationProvider;  
 }  
   
   
   
  
}

CONTROLLERS

AppointmentController: package com.themes.controllers;  
  
import org.springframework.beans.factory.annotation.Autowired;  
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.RestController;  
  
import com.themes.models.Appointment;  
import com.themes.services.AppointmentService;  
  
@RestController  
@RequestMapping("/apt")  
public class AppointmentController {  
  
 @Autowired  
 AppointmentService appointmentService;  
  
 @PostMapping("/create-appointment")  
 public Appointment saveAppointment(@RequestBody Appointment appointment) {  
 return appointmentService.createAppointmrnt(appointment);  
  
 }  
  
 @GetMapping("/getApt")  
 public Appointment getAppointment(Integer id) {  
 return appointmentService.getAppointmentWithId(id);  
 }  
  
 @GetMapping("/{userId}")  
 public Appointment getAppointmentWithAppointmentId(@PathVariable("userId") Integer id) {  
 return appointmentService.getAppointmentWithId(id);  
 }  
  
}

AuthController: package com.themes.controllers;  
  
import java.util.List;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.BadCredentialsException;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.GetMapping;  
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.RestController;  
  
import com.themes.models.JWTRequest;  
import com.themes.models.JWTResponse;  
import com.themes.models.User;  
import com.themes.repository.UserRepository;  
import com.themes.security.JWTHelper;  
import com.themes.services.UserService;  
  
@RestController  
@RequestMapping("/auth")  
public class AuthController {  
   
 @Autowired  
 private UserDetailsService userDetailsService;  
  
 @Autowired  
 private AuthenticationManager manager;  
   
 @Autowired  
 private UserService userService;  
   
   
 @Autowired  
 private JWTHelper helper;  
  
 private Logger logger = LoggerFactory.getLogger(AuthController.class);  
   
   
 @PostMapping("/login")  
 public ResponseEntity<JWTResponse> login(@RequestBody JWTRequest request) {  
  
 this.doAuthenticate(request.getEmail(), request.getPassword());  
  
  
 UserDetails userDetails = userDetailsService.loadUserByUsername(request.getEmail());  
 String token = this.helper.generateToken(userDetails);  
  
 JWTResponse response = new JWTResponse();   
 response.setToken(token);  
 response.setUsername(userDetails.getUsername());  
   
 return new ResponseEntity<>(response, HttpStatus.OK);  
 }  
  
 private void doAuthenticate(String email, String password) {  
  
 UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(email, password);  
 try {  
 manager.authenticate(authentication);  
  
  
 } catch (BadCredentialsException e) {  
 throw new BadCredentialsException(" Invalid Username or Password !!");  
 }  
  
 }  
  
 @ExceptionHandler(BadCredentialsException.class)  
 public String exceptionHandler() {  
 return "Credentials Invalid !!";  
 }  
   
 @PostMapping("/create-user")  
 public User createUser(@RequestBody User user)  
 {  
 return userService.createUser(user);  
 }  
   
 @GetMapping("/users")  
 public List<User> getAllUsers()  
 {  
 List<User> users = userService.getUsers();  
 return users;  
 }  
   
  
  
}

UserController: package com.themes.controllers;  
  
import java.util.Optional;  
  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import com.themes.models.User;  
import com.themes.services.UserService;  
  
@RestController  
@RequestMapping("/api")  
public class UserController {  
  
 UserService userService;  
   
   
 @PostMapping("/getUserWithEmail")  
 public Optional<User> getUserWithEmail(@RequestHeader String email)  
 {  
 return userService.getUserByEmail(email);  
 }  
   
}

UserResetPasswordController: package com.themes.controllers;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping("/reset-password")  
public class UserResetPasswordController {  
   
 @GetMapping("/reset")  
 public String getReset() {  
   
 return "reset successfully";  
   
 }  
  
}

UserUserNameAndPassword : package com.themes.controllers;  
  
public class UserUserNameAndPassword {  
 String email;  
 String password;  
   
   
 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;  
 }  
   
   
}

MODELS

Appointment : package com.themes.models;  
  
import java.sql.Date;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Appointment {  
   
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 Integer id;  
   
 private String aFirstname;  
 private String aLastname;  
 private String aEmail;  
 private String aPhoneNumber;  
 private String aCity;  
 private String aState;  
 private String aStreetAddress;  
 private String aZipCode;  
 private Date createdDate;  
 private String aWorkingTime;  
 private String aWorkType;  
 private String status;  
 private Integer aUserId;  
   
}

ContractorDetails : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
  
@Entity  
public class ContractorDetails {  
   
 @Id  
 @GeneratedValue(strategy=GenerationType.*AUTO*)  
 private int id;  
   
 private String gender;  
   
 private String aadharIdNo;  
   
 private String username;  
   
 private String flatNo;  
   
 private String city;  
   
 private String state;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getGender() {  
 return gender;  
 }  
  
 public void setGender(String gender) {  
 this.gender = gender;  
 }  
  
 public String getAadharIdNo() {  
 return aadharIdNo;  
 }  
  
 public void setAadharIdNo(String aadharIdNo) {  
 this.aadharIdNo = aadharIdNo;  
 }  
  
 public String getFlatNo() {  
 return flatNo;  
 }  
  
 public void setFlatNo(String flatNo) {  
 this.flatNo = flatNo;  
 }  
  
 public String getCity() {  
 return city;  
 }  
   
   
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getState() {  
 return state;  
 }  
  
 public void setState(String state) {  
 this.state = state;  
 }  
}

JWTRequest : package com.themes.models;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
import lombok.ToString;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
@ToString  
public class JWTRequest {  
   
 private String email;  
 private String password;  
 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;  
 }

JWTResponse : package com.themes.models;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
import lombok.ToString;  
  
@Getter  
@Setter  
  
@Builder  
@ToString  
public class JWTResponse {  
   
 String username;  
 String token;  
   
   
   
 public JWTResponse() {  
 super();  
 }  
 public JWTResponse(String username, String token) {  
 super();  
 this.username = username;  
 this.token = token;  
 }  
 public String getUsername() {  
 return username;  
 }  
 public void setUsername(String username) {  
 this.username = username;  
 }  
 public String getToken() {  
 return token;  
 }  
 public void setToken(String token) {  
 this.token = token;  
 }

:Project : package com.themes.models;  
  
import java.sql.Date;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Project {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer projectId;  
   
 private Date pCreatedDate;  
 private String pSerialNo;  
 private Integer userId;  
   
   
   
   
 public Project() {  
 // *TODO Auto-generated constructor stub* }  
  
  
  
  
 public Project(Integer projectId, Date pCreatedDate, String pSerialNo, Integer userId) {  
 super();  
 this.projectId = projectId;  
 this.pCreatedDate = pCreatedDate;  
 this.pSerialNo = pSerialNo;  
 this.userId = userId;  
 }  
  
  
  
  
 public Integer getProjectId() {  
 return projectId;  
 }  
  
  
  
  
 public void setProjectId(Integer projectId) {  
 this.projectId = projectId;  
 }  
  
  
  
  
 public Date getpCreatedDate() {  
 return pCreatedDate;  
 }  
  
  
  
  
 public void setpCreatedDate(Date pCreatedDate) {  
 this.pCreatedDate = pCreatedDate;  
 }  
  
  
  
  
 public String getpSerialNo() {  
 return pSerialNo;  
 }  
  
  
  
  
 public void setpSerialNo(String pSerialNo) {  
 this.pSerialNo = pSerialNo;  
 }  
  
  
  
  
 public Integer getUserId() {  
 return userId;  
 }  
  
  
  
  
 public void setUserId(Integer userId) {  
 this.userId = userId;  
 }

ProjectAddress Details : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class ProjectAddressDetails {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer id;  
   
 private String address1;  
 private String address2;  
 private String city;  
 private String pinCode;  
 private String latitude;  
 private String longitude;  
 private Integer projectId;  
   
 public ProjectAddressDetails() {  
 // *TODO Auto-generated constructor stub* }  
  
 public ProjectAddressDetails(Integer id, String address1, String address2, String city, String pinCode,  
 String latitude, String longitude, Integer projectId) {  
 super();  
 this.id = id;  
 this.address1 = address1;  
 this.address2 = address2;  
 this.city = city;  
 this.pinCode = pinCode;  
 this.latitude = latitude;  
 this.longitude = longitude;  
 this.projectId = projectId;  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getAddress1() {  
 return address1;  
 }  
  
 public void setAddress1(String address1) {  
 this.address1 = address1;  
 }  
  
 public String getAddress2() {  
 return address2;  
 }  
  
 public void setAddress2(String address2) {  
 this.address2 = address2;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getPinCode() {  
 return pinCode;  
 }  
  
 public void setPinCode(String pinCode) {  
 this.pinCode = pinCode;  
 }  
  
 public String getLatitude() {  
 return latitude;  
 }  
  
 public void setLatitude(String latitude) {  
 this.latitude = latitude;  
 }  
  
 public String getLongitude() {  
 return longitude;  
 }  
  
 public void setLongitude(String longitude) {  
 this.longitude = longitude;  
 }  
  
 public Integer getProjectId() {  
 return projectId;  
 }  
  
 public void setProjectId(Integer projectId) {  
 this.projectId = projectId;  
 }

ProjectServiceDetails : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
  
@Entity  
public class ProjectServiceDetails {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer projectServiceId;  
 private Integer serviceId;  
 private Integer projectId;  
   
   
 public ProjectServiceDetails() {  
 // *TODO Auto-generated constructor stub* }  
  
  
 public ProjectServiceDetails(Integer projectServiceId, Integer serviceId, Integer projectId) {  
 super();  
 this.projectServiceId = projectServiceId;  
 this.serviceId = serviceId;  
 this.projectId = projectId;  
 }  
  
  
 public Integer getProjectServiceId() {  
 return projectServiceId;  
 }  
  
  
 public void setProjectServiceId(Integer projectServiceId) {  
 this.projectServiceId = projectServiceId;  
 }  
  
  
 public Integer getServiceId() {  
 return serviceId;  
 }  
  
  
 public void setServiceId(Integer serviceId) {  
 this.serviceId = serviceId;  
 }  
  
  
 public Integer getProjectId() {  
 return projectId;  
 }  
  
  
 public void setProjectId(Integer projectId) {  
 this.projectId = projectId;  
 }  
  
  
   
}

ProjectWorkerDetails : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class ProjectWorkerDetails {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer id;  
 private Integer workerId;  
 private Integer ProjectId;  
   
   
 public ProjectWorkerDetails() {  
 // *TODO Auto-generated constructor stub* }  
  
  
 public ProjectWorkerDetails(Integer id, Integer workerId, Integer projectId) {  
 super();  
 this.id = id;  
 this.workerId = workerId;  
 ProjectId = projectId;  
 }  
  
  
 public Integer getId() {  
 return id;  
 }  
  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
  
 public Integer getWorkerId() {  
 return workerId;  
 }  
  
  
 public void setWorkerId(Integer workerId) {  
 this.workerId = workerId;  
 }  
  
  
 public Integer getProjectId() {  
 return ProjectId;  
 }  
  
  
 public void setProjectId(Integer projectId) {  
 ProjectId = projectId;  
 }  
   
   
  
}

Service : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Service {  
  
   
   
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer serviceId;  
   
 private String serviceName;  
   
 public Service() {  
 // *TODO Auto-generated constructor stub* }  
  
 public Service(Integer serviceId, String serviceName) {  
 super();  
 this.serviceId = serviceId;  
 this.serviceName = serviceName;  
 }  
  
 public Integer getServiceId() {  
 return serviceId;  
 }  
  
 public void setServiceId(Integer serviceId) {  
 this.serviceId = serviceId;  
 }  
  
 public String getServiceName() {  
 return serviceName;  
 }  
  
 public void setServiceName(String serviceName) {  
 this.serviceName = serviceName;  
 }  
   
   
   
  
}

Skill : package com.themes.models;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Skill {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private int skillId;  
   
 private String skillName;  
   
   
   
  
 public Skill() {  
 super();  
 }  
  
 public Skill(int skillId, String skillName) {  
 super();  
 this.skillId = skillId;  
 this.skillName = skillName;  
 }  
  
 public int getSkillId() {  
 return skillId;  
 }  
  
 public void setSkillId(int skillId) {  
 this.skillId = skillId;  
 }  
  
 public String getSkillName() {  
 return skillName;  
 }  
  
 public void setSkillName(String skillName) {  
 this.skillName = skillName;  
 }  
   
   
   
   
   
   
}

User : package com.themes.models;  
  
import java.util.Collection;  
  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class User implements UserDetails{  
   
 */\*\*  
 \*   
 \*/* private static final long *serialVersionUID* = 1L;  
  
 @Id  
 @GeneratedValue(strategy=GenerationType.*AUTO*)  
 private Integer id;  
   
 private String firstname;  
   
 private String lastname;  
   
 private String phone;  
   
 private String email;  
   
 private String password;  
   
 private String username;  
   
 private String createdDate;  
   
 private String userType;  
   
   
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public String getUsername() {  
 return this.email;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
   
   
  
 public String getCreatedDate() {  
 return createdDate;  
 }  
  
 public void setCreatedDate(String createdDate) {  
 this.createdDate = createdDate;  
 }  
  
 public String getUserType() {  
 return userType;  
 }  
  
 public void setUserType(String userType) {  
 this.userType = userType;  
 }  
   
   
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
   
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
   
   
 return null;  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 // *TODO Auto-generated method stub* return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 // *TODO Auto-generated method stub* return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 // *TODO Auto-generated method stub* return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 // *TODO Auto-generated method stub* return true;  
 }  
   
   
   
  
}

UserWithDetails : package com.themes.models;  
  
  
  
public class UserWithDetails {  
   
   
 private String username;  
 private String city;  
   
   
   
   
  
 public UserWithDetails(String username, String city) {  
 super();  
 this.username = username;  
 this.city = city;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
   
   
   
}

REPOSITORY

AppointmentRepository : package com.themes.repository;  
  
import java.util.List;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import com.themes.models.Appointment;  
import com.themes.models.User;  
  
public interface AppointmentRepository extends JpaRepository<Appointment,Integer>{  
  
 @Query(value="select \* from appointment", nativeQuery=true)  
 public List<Appointment> getAppointment();  
   
   
 @Query(value="SELECT \* FROM themes\_interior\_db.appointment where id=1", nativeQuery=true)  
 public Appointment getAppointmentWithId();  
   
 @Query(value="SELECT \* FROM themes\_interior\_db.appointment apt where apt.id=:id", nativeQuery=true)  
 public Appointment getAppointmentWithAppointmentId(@Param("id") Integer id);  
   
   
}

UserRepository : package com.themes.repository;  
  
import java.util.List;  
import java.util.Optional;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import org.springframework.web.bind.annotation.RequestParam;  
import com.themes.models.User;  
import com.themes.models.UserWithDetails;  
  
  
  
  
public interface UserRepository extends JpaRepository<User,Integer>{  
   
   
   
 public Optional<User> findByEmail(String email);  
   
 public Optional<User> findByEmailAndPassword(String email,String password);  
   
   
   
}

SECURITY

JWTAuthenticationEntryPoint : package com.themes.security;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.web.AuthenticationEntryPoint;  
import org.springframework.stereotype.Component;  
  
import jakarta.servlet.ServletException;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
  
@Component  
public class JWTAuthenticationEntryPoint implements AuthenticationEntryPoint {  
  
 @Override  
 public void commence(HttpServletRequest request, HttpServletResponse response,  
 AuthenticationException authException) throws IOException, ServletException {  
 response.setStatus(HttpServletResponse.*SC\_UNAUTHORIZED*);  
 PrintWriter writer = response.getWriter();  
   
 writer.println("Access Denied!"+authException.getMessage());  
   
 }  
  
}

JWTAuthenticationFilter : package com.themes.security;  
  
import java.io.IOException;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
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 io.jsonwebtoken.ExpiredJwtException;  
import io.jsonwebtoken.MalformedJwtException;  
import jakarta.servlet.FilterChain;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
  
@Component  
public class JWTAuthenticationFilter extends OncePerRequestFilter{  
   
   
 private Logger logger = LoggerFactory.*getLogger*(OncePerRequestFilter.class);  
 @Autowired  
 private JWTHelper jwtHelper;  
  
  
 @Autowired  
 private UserDetailsService userDetailsService;  
  
 @Override  
 protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {  
  
// try {  
// Thread.sleep(500);  
// } catch (InterruptedException e) {  
// throw new RuntimeException(e);  
// }  
 //Authorization  
  
 String requestHeader = request.getHeader("Authorization");  
 //Bearer 2352345235sdfrsfgsdfsdf  
 logger.info(" Header : {}", requestHeader);  
 String username = null;  
 String token = null;  
 if (requestHeader != null && requestHeader.startsWith("Bearer")) {  
 //looking good  
 token = requestHeader.substring(7);  
 try {  
  
 username = this.jwtHelper.getUsernameFromToken(token);  
  
 } catch (IllegalArgumentException e) {  
 logger.info("Illegal Argument while fetching the username !!");  
 e.printStackTrace();  
 } catch (ExpiredJwtException e) {  
 logger.info("Given jwt token is expired !!");  
 e.printStackTrace();  
 } catch (MalformedJwtException e) {  
 logger.info("Some changed has done in token !! Invalid Token");  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
  
 }  
  
  
 } else {  
 logger.info("Invalid Header Value !! ");  
 }  
  
  
 //  
 if (username != null && SecurityContextHolder.*getContext*().getAuthentication() == null) {  
  
  
 //fetch user detail from username  
 UserDetails userDetails = this.userDetailsService.loadUserByUsername(username);  
 Boolean validateToken = this.jwtHelper.validateToken(token, userDetails);  
 if (validateToken) {  
  
 //set the authentication  
 UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());  
 authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));  
 SecurityContextHolder.*getContext*().setAuthentication(authentication);  
  
  
 } else {  
 logger.info("Validation fails !!");  
 }  
  
  
 }  
  
 filterChain.doFilter(request, response);  
  
  
 }  
   
   
  
}

JWTHelper: package com.themes.security;  
  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.function.Function;  
  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.stereotype.Component;  
  
import io.jsonwebtoken.Claims;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
  
@Component  
public class JWTHelper {  
  
   
 //requirement :  
 public static final long *JWT\_TOKEN\_VALIDITY* = 5 \* 60 \* 60;  
  
 // public static final long JWT\_TOKEN\_VALIDITY = 60;  
 private String secret = "afafasfafafasfasfasfafacasdasfasxASFACASDFACASDFASFASFDAFASFASDAADSCSDFADCVSGCFVADXCcadwavfsfarvf";  
  
 //retrieve username from jwt token  
 public String getUsernameFromToken(String token) {  
 return getClaimFromToken(token, Claims::getSubject);  
 }  
  
 //retrieve expiration date from jwt token  
 public Date getExpirationDateFromToken(String token) {  
 return getClaimFromToken(token, Claims::getExpiration);  
 }  
  
 public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver) {  
 final Claims claims = getAllClaimsFromToken(token);  
 return claimsResolver.apply(claims);  
 }  
  
 //for retrieveing any information from token we will need the secret key  
 private Claims getAllClaimsFromToken(String token) {  
 return Jwts.*parser*().setSigningKey(secret).parseClaimsJws(token).getBody();  
 }  
  
 //check if the token has expired  
 private Boolean isTokenExpired(String token) {  
 final Date expiration = getExpirationDateFromToken(token);  
 return expiration.before(new Date());  
 }  
  
 //generate token for user  
 public String generateToken(UserDetails userDetails) {  
 Map<String, Object> claims = new HashMap<>();  
 return doGenerateToken(claims, userDetails.getUsername());  
 }  
  
 //while creating the token -  
 //1. Define claims of the token, like Issuer, Expiration, Subject, and the ID  
 //2. Sign the JWT using the HS512 algorithm and secret key.  
 //3. According to JWS Compact Serialization(https://tools.ietf.org/html/draft-ietf-jose-json-web-signature-41#section-3.1)  
 // compaction of the JWT to a URL-safe string  
 private String doGenerateToken(Map<String, Object> claims, String subject) {  
  
 return Jwts.*builder*().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + *JWT\_TOKEN\_VALIDITY* \* 1000))  
 .signWith(SignatureAlgorithm.*HS512*, secret).compact();  
 }  
  
 //validate token  
 public Boolean validateToken(String token, UserDetails userDetails) {  
 final String username = getUsernameFromToken(token);  
 return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));  
 }  
  
   
}

SERVICES

=IMPL

AppointmentServicelmpl : package com.themes.services.impl;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import com.themes.models.Appointment;  
import com.themes.repository.AppointmentRepository;  
import com.themes.services.AppointmentService;  
  
  
@Service  
public class AppointmentServiceImpl implements AppointmentService{  
   
 @Autowired  
 AppointmentRepository appointmentRepository;  
  
 public AppointmentServiceImpl() {  
 // *TODO Auto-generated constructor stub* }  
  
 @Override  
 public Appointment createAppointmrnt(Appointment appointment) {  
 // *TODO Auto-generated method stub* return appointmentRepository.save(appointment);  
 }  
  
 @Override  
 public Appointment getAppointmentWithId(Integer id) {  
 // *TODO Auto-generated method stub* return appointmentRepository.getAppointmentWithAppointmentId(id);  
 }  
  
   
  
}

CustomUserDetails Service :

AppointmentService : package com.themes.services;  
  
import org.springframework.stereotype.Service;  
  
import com.themes.models.Appointment;  
  
@Service  
public interface AppointmentService {  
  
 public Appointment createAppointmrnt(Appointment appointment);  
   
 public Appointment getAppointmentWithId(Integer id);  
  
   
   
}

CustomUserDetails Service : package com.themes.services;  
  
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.themes.models.User;  
import com.themes.repository.UserRepository;  
  
@Service  
public class CustomUserDetailsService implements UserDetailsService{  
   
 @Autowired  
 private UserRepository userRepository;  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 // *TODO Auto-generated method stub* User user = userRepository.findByEmail(username).orElseThrow(() -> new RuntimeException("User not found"));  
   
 return user;  
 }  
  
}

ProjectService: package com.themes.services;  
  
import java.util.List;  
  
import org.springframework.stereotype.Service;  
  
import com.themes.models.Project;  
import com.themes.models.ProjectAddressDetails;  
import com.themes.models.ProjectServiceDetails;  
  
@Service  
public interface ProjectService {  
  
 public Project createProject(Project project);  
   
   
 public Project getProjet(Integer projectId);  
   
   
 public ProjectAddressDetails getProjectAddress(Integer projectId);  
   
 public List<ProjectServiceDetails> getProjectService(Integer projectId);  
   
   
}

SendMailService : package com.themes.services;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.mail.SimpleMailMessage;  
import org.springframework.mail.javamail.JavaMailSender;  
import org.springframework.stereotype.Service;  
  
@Service  
public class SendMailService {  
  
 @Autowired  
 private JavaMailSender mailSender;  
   
   
   
 public void sendEmail(String toMail, String subject, String body)  
 {  
 SimpleMailMessage mailMessage = new SimpleMailMessage();  
   
 mailMessage.setFrom("ritik.masters.mca@gmail.com");  
 mailMessage.setTo(toMail);  
 mailMessage.setSubject(subject);  
 mailMessage.setText(body);  
   
 mailSender.send(mailMessage);  
   
 System.*out*.println("Mail Sent successfully");  
   
   
   
 }  
   
}

UserService : package com.themes.services;  
  
import java.util.List;  
import java.util.Optional;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import com.themes.models.User;  
import com.themes.repository.UserRepository;  
  
@Service  
public class UserService {  
  
   
 @Autowired  
 private UserRepository userRepository;  
   
 @Autowired  
 private PasswordEncoder passwordEncoder;  
   
   
 public List<User> getUsers()  
 {  
 return userRepository.findAll();  
 }  
   
   
 public Optional<User> getUserByEmail(String email)  
 {  
   
   
 return userRepository.findByEmail(email);  
 }  
   
 public User createUser(User user)  
 {  
   
   
 String username = user.getEmail();  
   
 user.setUsername(username);  
   
   
   
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
   
 return userRepository.save(user);  
 }  
}

Themesoneaaplication

:

package com.themes;  
  
import java.util.List;  
import java.util.Random;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.context.event.ApplicationReadyEvent;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.event.EventListener;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.web.servlet.config.annotation.CorsRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import com.themes.models.User;  
import com.themes.repository.UserRepository;  
import com.themes.services.SendMailService;  
  
  
  
  
  
@SpringBootApplication  
public class ThemesOneApplication {   
  
 @Autowired  
 private SendMailService sendMail;  
   
   
   
 public static void main(String[] args) {  
 SpringApplication.*run*(ThemesOneApplication.class, args);   
   
   
 }  
   
 @EventListener(ApplicationReadyEvent.class)  
 public void send()  
 {   
   
  
   
 Random rnd = new Random();  
 int number = rnd.nextInt(999999);  
   
   
 //String body = String.format("%06d", number);  
 System.*out*.println("Application started...");  
   
 }  
   
   
 @Bean  
 WebMvcConfigurer corsConfigurer() {  
 return new WebMvcConfigurer() {  
 public void addCorsMappings(CorsRegistry registry) {  
 registry.addMapping("/\*\*").allowedOrigins("http://localhost:4200");  
 }  
 };  
 }   
   
}

themes-one D:\project\JAVA PROJECT\themes-one

=

src

=

CONFIG

© AppConfig

SecurityConfig

CONTROLLERS

AppointmentController

AuthController

UserController

UserResetPasswordController

UserUserNameAndPassword

MODELS

Appointment

ContractorDetails

JWTRequest

JWTResponse

© Project

ProjectAddress Details

ProjectServiceDetails

ProjectWorkerDetails

Service

Skill

User

UserWithDetails

REPOSITORY

AppointmentRepository

UserRepository

SECURITY

JWTAuthenticationEntryPoint

JWTAuthenticationFilter

JWTHelper

SERVICES

=IMPL

AppointmentServicelmpl

CustomUserDetails Service

AppointmentService

CustomUserDetails Service

ProjectService

SendMailService

UserService

Themesoneaaplication