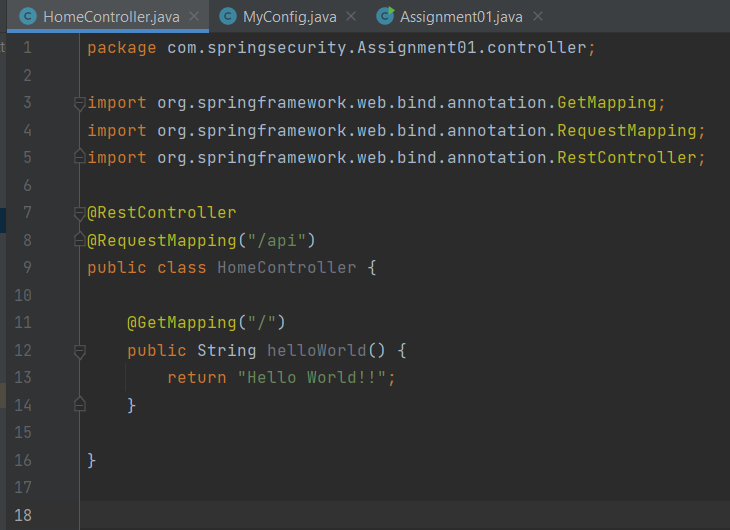
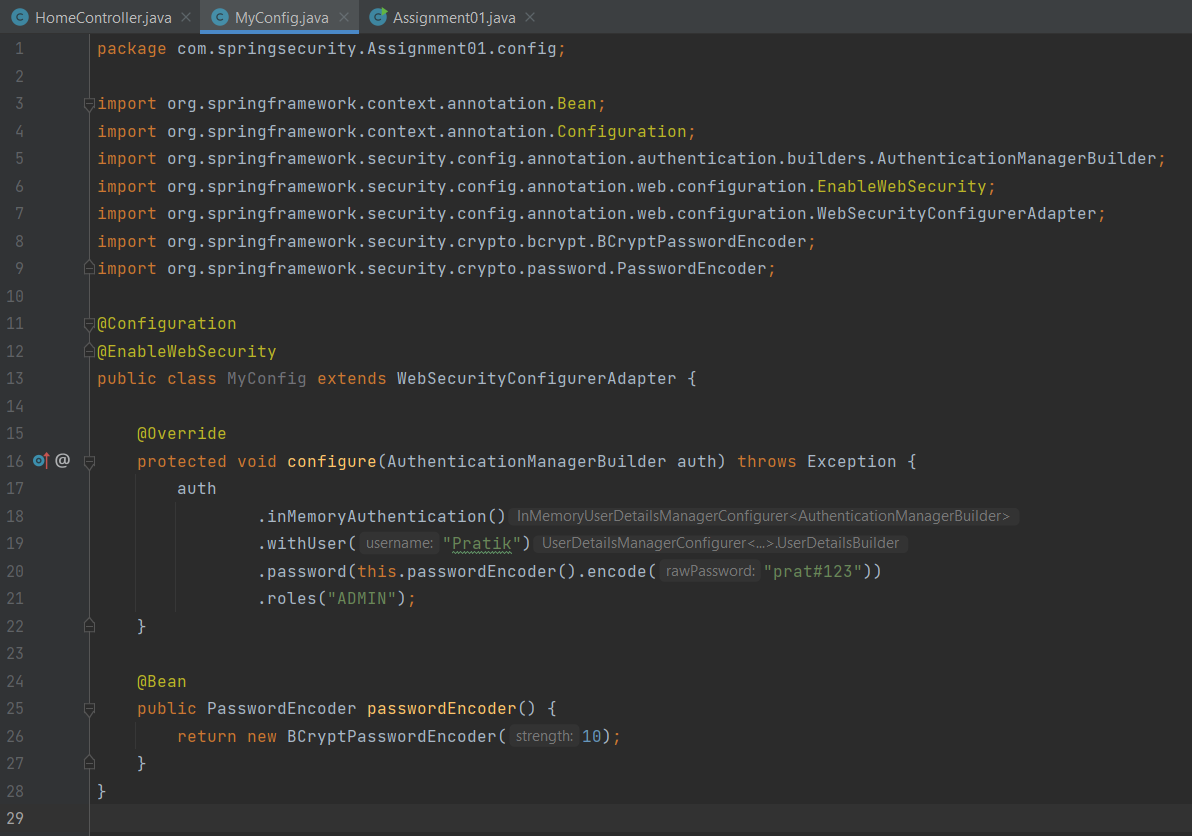
**Spring Security Assignment 01**

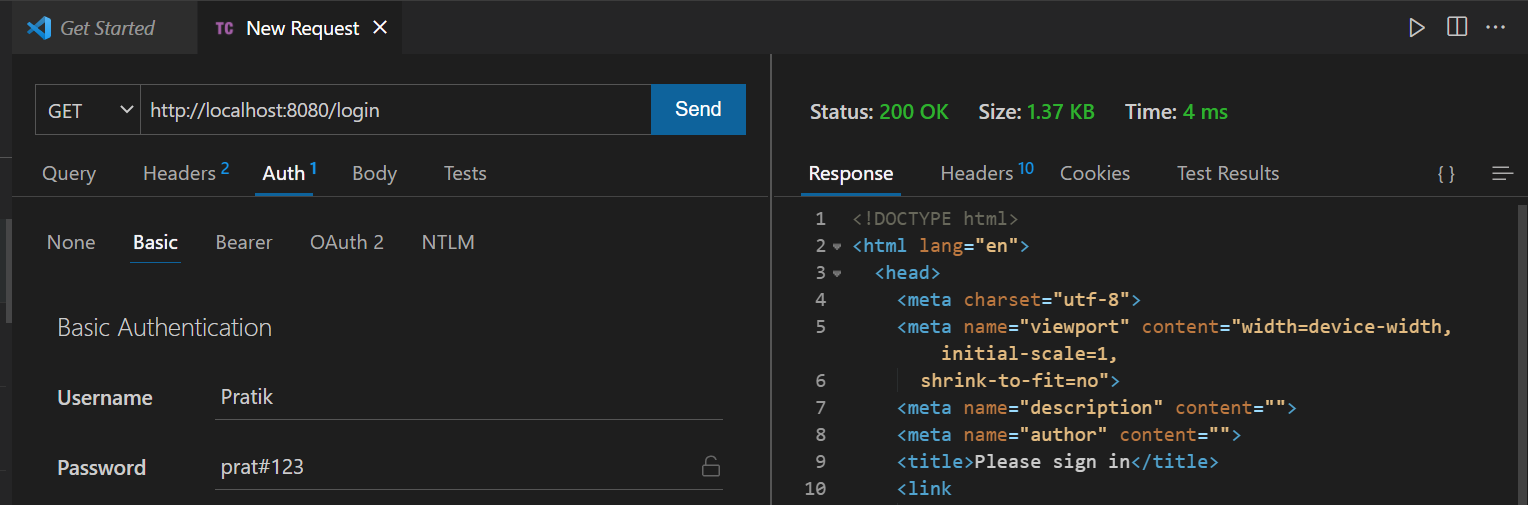
1) Design and develop a Spring security Hello World application by using default login form provided by spring security to secure URL access say, for example to access the content of an “admin” page, user needs to enter valid credentials. User must also logged out if successfully logged in. Use Java Based and annotation based configuration and In-memory authentication.

**Controller.java**

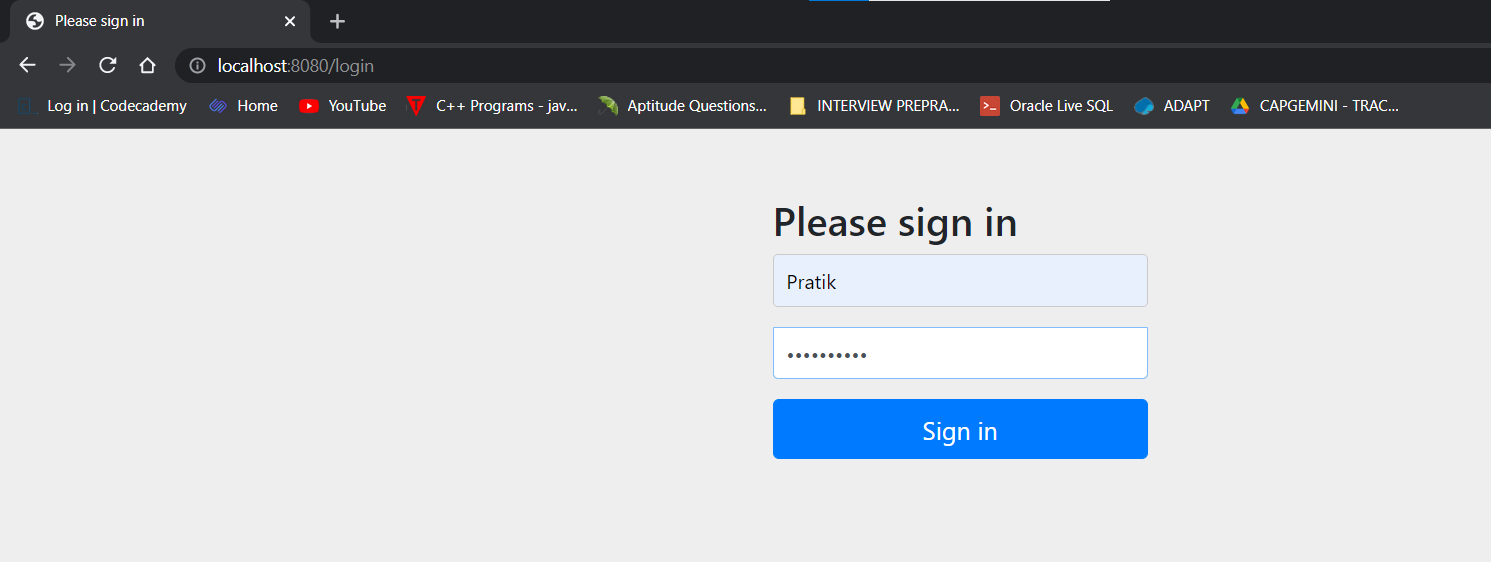


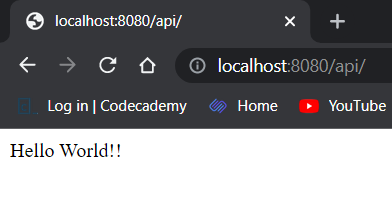


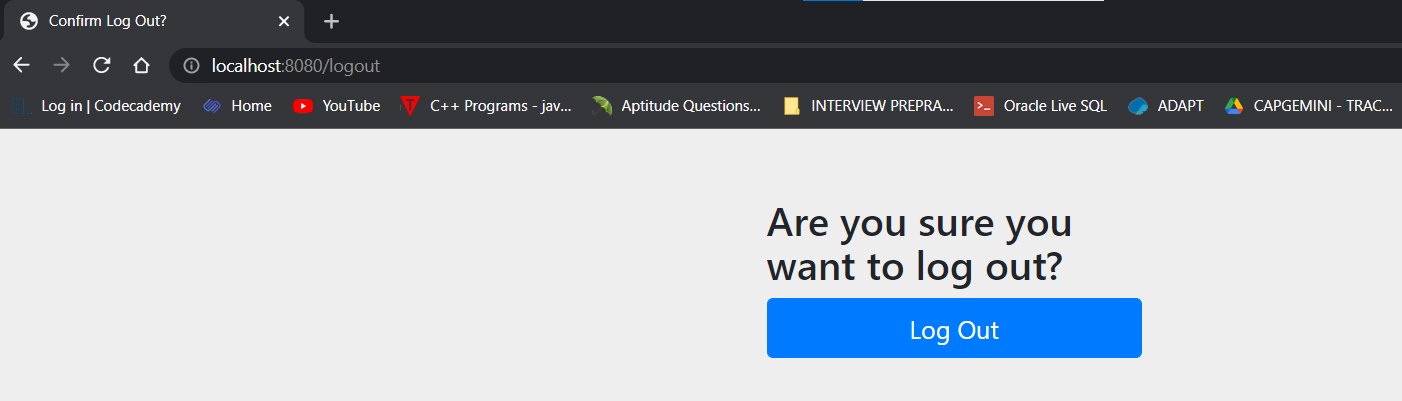
API Testing using Thunder Client

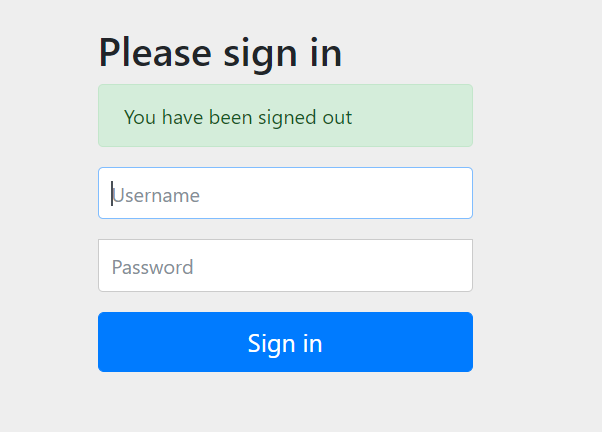


Output :-

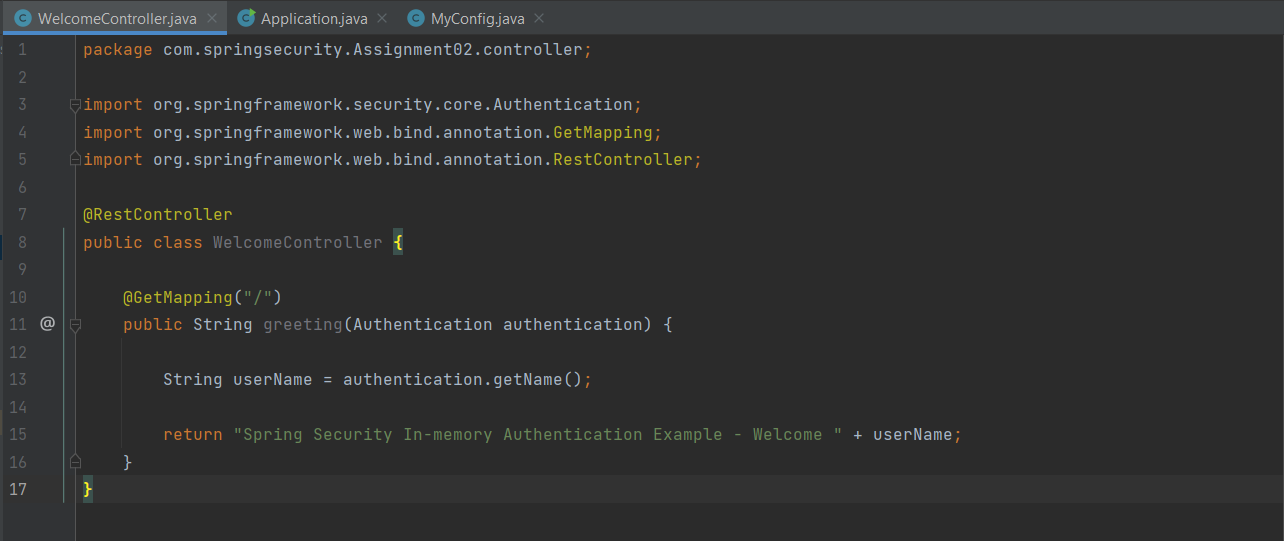


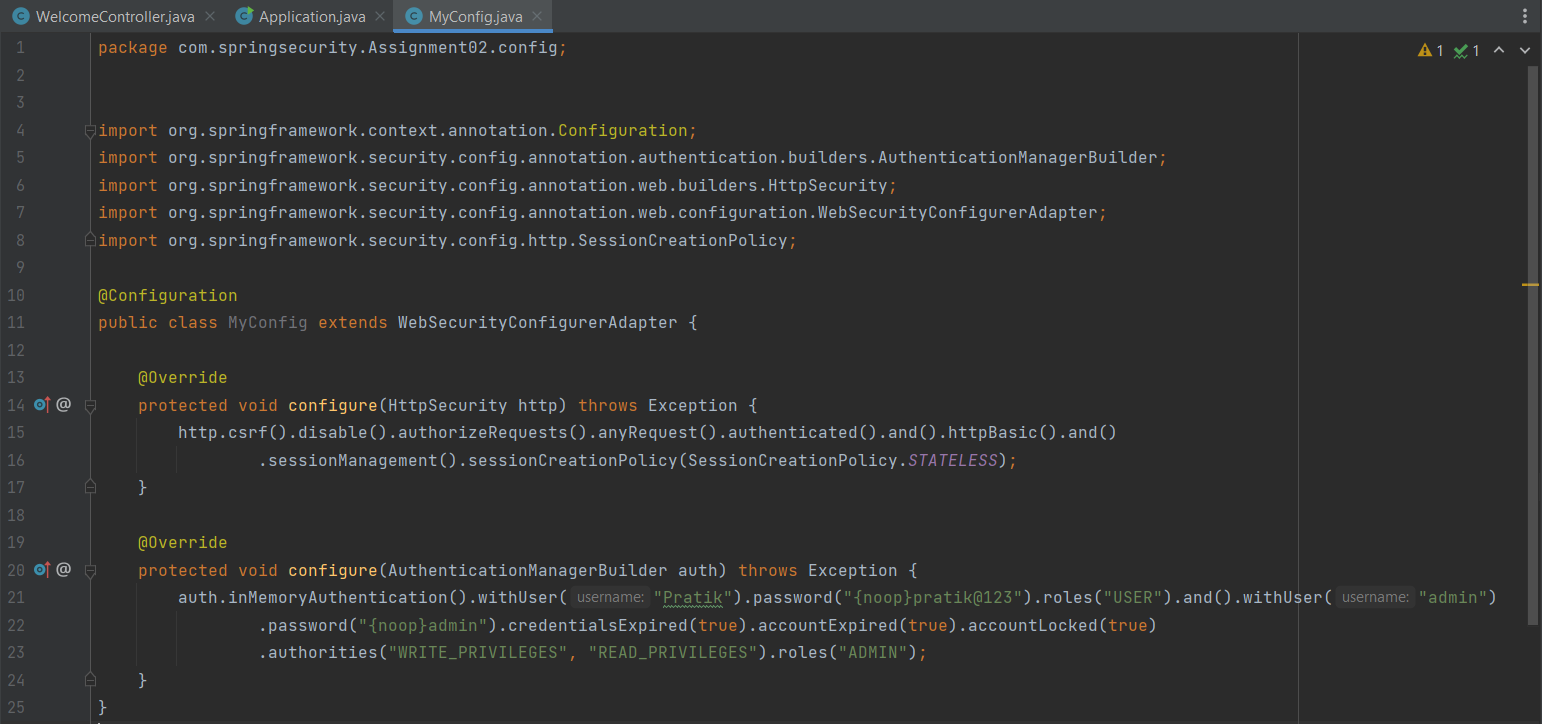


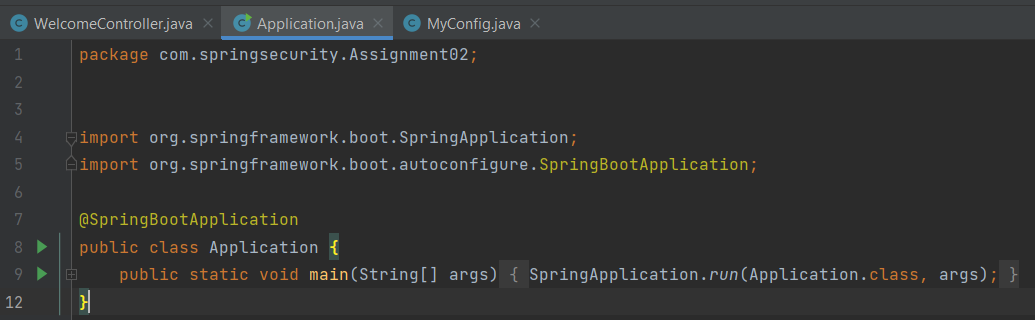


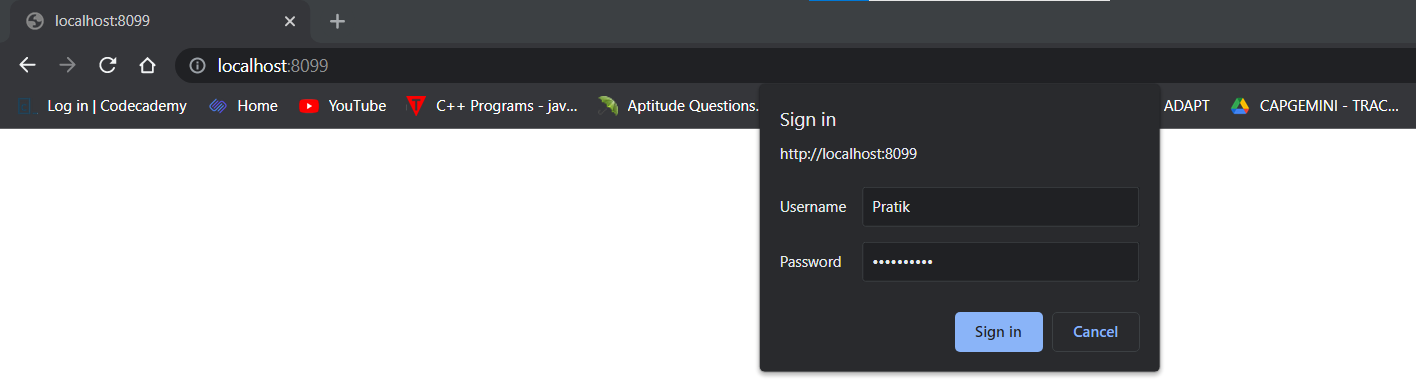


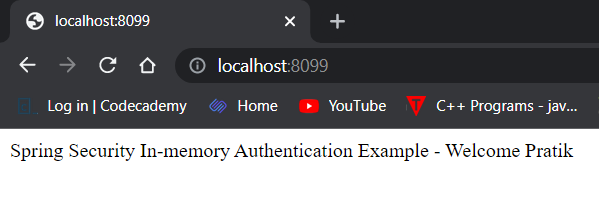
2) Modify the above application to use custom login form instead of default login form provided by spring security. Use Java Based and annotation based configuration and In-memory authentication.







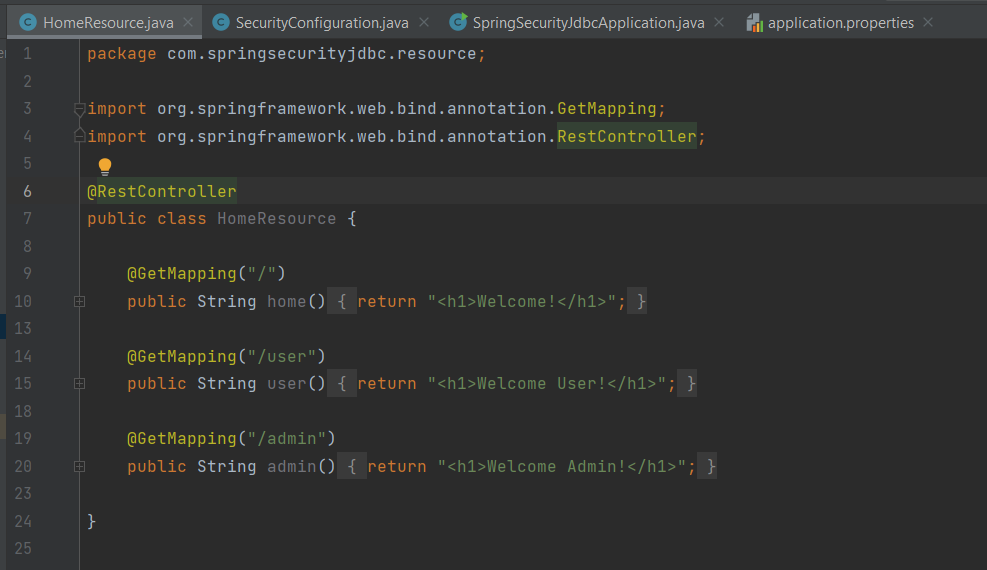




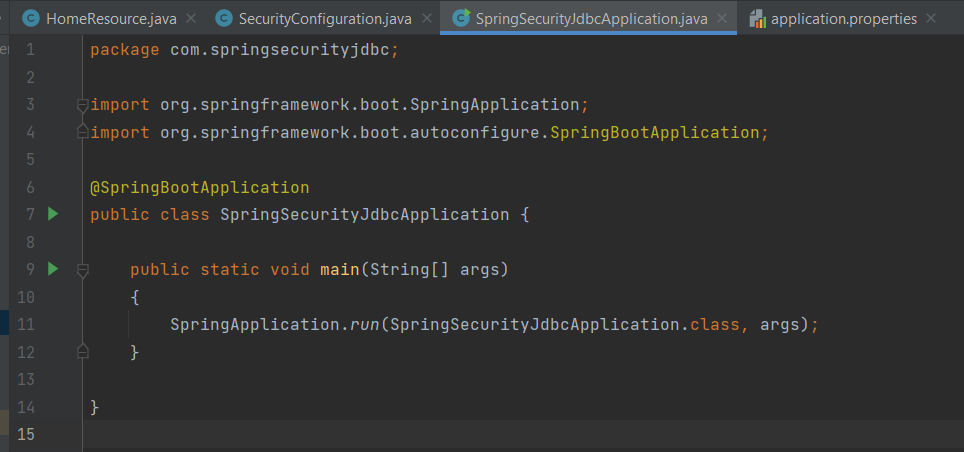
3) Modify the above application and use database authentication using JDBC instead of In memory authentication.

Use Java Based and annotation based configuration and In-memory authentication.

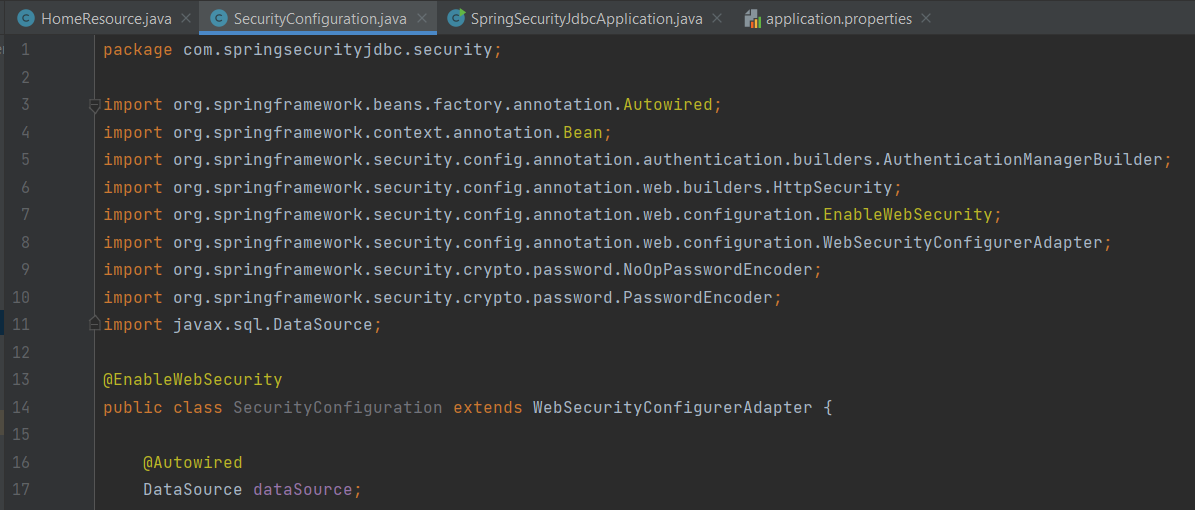
**HomeResource.java**

****

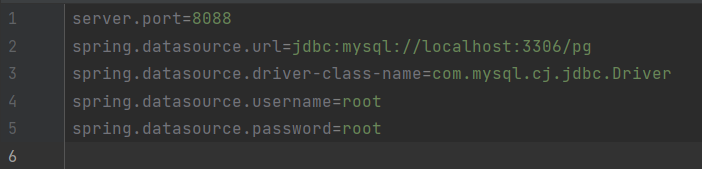
**SpringSecurityJdbcApplication.java**

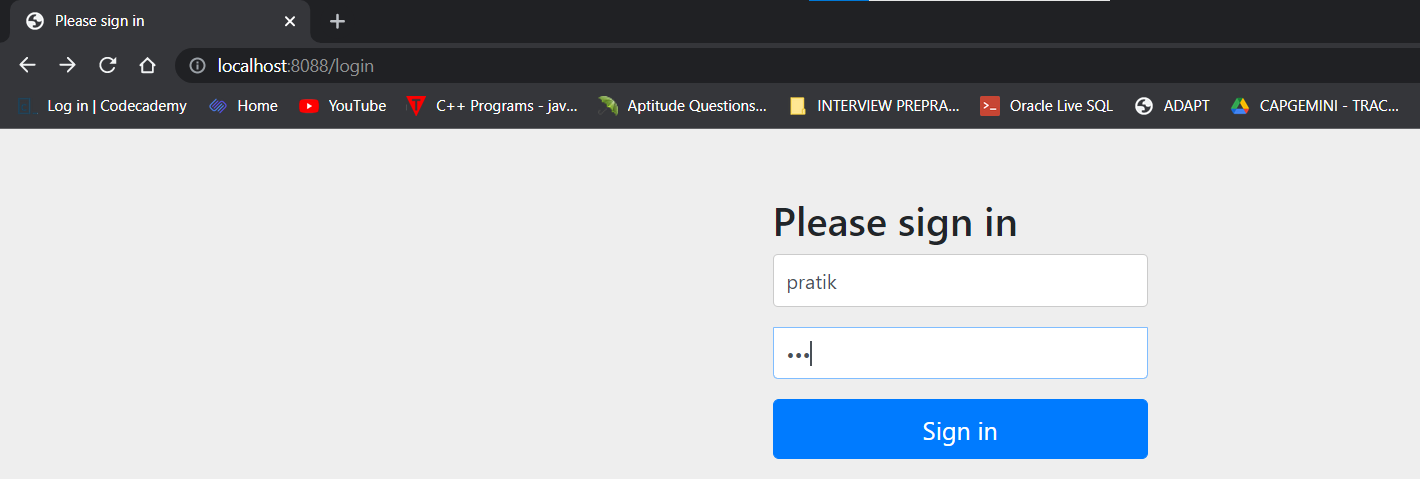


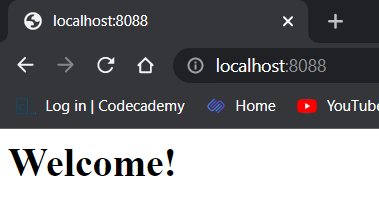
**SecurityConfiguration.java**



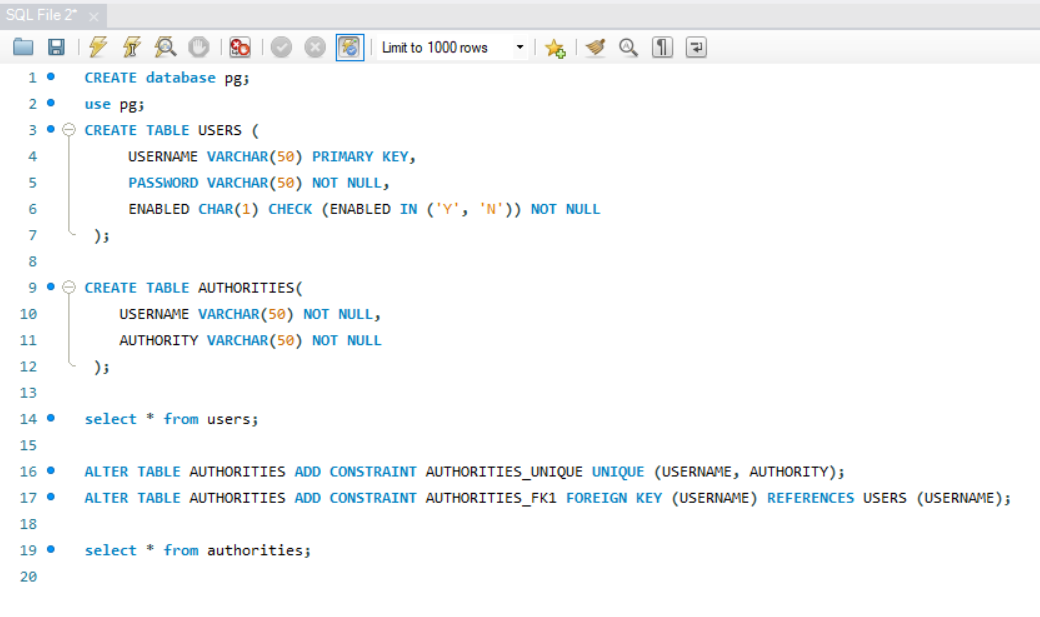


**application.properties**





**MySql Databse :-**



4) Modify the above application to limit the number of login attempts.

**SpringSecurityConfig.java**

package SpringSecurity.Assignment04.configuration;  
  
import SpringSecurity.Assignment04.services.CustomLoginFailureHandler;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
  
@EnableWebSecurity  
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Autowired  
 UserDetailsService userDetailsService;  
  
 @Autowired  
 CustomLoginFailureHandler customLoginFailureHandler;  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userDetailsService);  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/admin").hasRole("ADMIN")  
 .antMatchers("/user").hasAnyRole("USER","ADMIN")  
 .antMatchers("/").permitAll()  
 .antMatchers("/login\*\*").permitAll()  
 .and().formLogin()  
 .loginPage("/login")  
 .permitAll()

.failureHandler(customLoginFailureHandler)  
 .permitAll()  
 .and()  
 .logout().permitAll()  
 .and().rememberMe();  
 }  
  
 @Bean  
 public PasswordEncoder getPasswordEncoder(){  
 return new BCryptPasswordEncoder();  
 }  
}

**HomeController.java**

package SpringSecurity.Assignment04.controller;  
  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HomeController {  
 @GetMapping("/")  
 public String home(){  
 return ("<h1>Welcome</h1>");  
 }  
 @GetMapping("/user")  
 public String user(){  
 return ("<h1>Welcome user</h1>");  
 }  
 @GetMapping("/admin")  
 public String admin(){  
 return ("<h1>Welcome admin</h1>");  
 }  
}

**LoginController.java**

package SpringSecurity.Assignment04.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;

@Controller  
public class LoginController {

@RequestMapping(value = "/login", method = RequestMethod.*GET*)  
 public String login(Model model, String error, String logout) {  
 if (error != null)  
 model.addAttribute("errorMsg", "Your username and password are invalid.");  
  
 if (logout != null)  
 model.addAttribute("msg", "You have been logged out successfully.");  
  
 return "login";  
 }  
  
 @RequestMapping(value = "/failure")  
 public String failure(Model model){  
  
 model.addAttribute("loginError",true);  
 model.addAttribute("exception",true);  
 return "login";  
 }  
   
}

**MyUserDetails.java**

package SpringSecurity.Assignment04.Model;  
  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import java.util.Arrays;  
import java.util.Collection;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class MyUserDetails implements UserDetails {  
  
 private String userName;  
 private String password;  
 private boolean active;  
 private String roles;  
 private boolean accountNonLocked;  
 private List<GrantedAuthority> authorities;

public MyUserDetails(User user){  
 System.*out*.println("Inside my user details");  
 this.userName=user.getUserName();  
 this.password=user.getPassword();  
 this.active=user.isActive();  
 this.accountNonLocked=user.isAccountNonLocked();  
 this. authorities= Arrays.*stream*(user.getRoles().split(","))  
 .map(SimpleGrantedAuthority::new)  
 .collect(Collectors.*toList*());  
 System.*out*.println(userName);  
 System.*out*.println(password);  
 System.*out*.println(active);  
 System.*out*.println(authorities);  
  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
 return authorities;  
 }  
  
 @Override  
 public String getPassword() {  
 return password;  
 }  
  
 @Override  
 public String getUsername() {  
 return userName;  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 return accountNonLocked;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 return active;  
 }  
}

**User.java**

package SpringSecurity.Assignment04.Model;  
  
import javax.persistence.\*;  
import java.util.Date;  
  
@Entity  
@Table(name = "user")  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private int id;  
 private String username;  
 private String password;  
 private boolean active;  
 private String roles;  
  
 public User(){}  
  
 public User(int id, String username, String password, boolean active, String roles, boolean accountNonLocked, int failedAttempt, Date lockTime) {  
 this.id = id;  
 this.username = username;  
 this.password = password;  
 this.active = active;  
 this.roles = roles;  
 this.accountNonLocked = accountNonLocked;  
 this.failedAttempt = failedAttempt;  
 this.lockTime = lockTime;  
 }  
  
 @Column(name = "account\_no\_locker")  
 private boolean accountNonLocked;  
  
 @Column(name = "failed\_attempt")  
 private int failedAttempt;  
  
 @Column(name = "lock\_time")  
 private Date lockTime;  
   
 public boolean isAccountNonLocked() {  
 return accountNonLocked;  
 }  
  
 public void setAccountNonLocked(boolean accountNonLocked) {  
 this.accountNonLocked = accountNonLocked;  
 }  
  
 public int getFailedAttempt() {  
 return failedAttempt;  
 }

public void setFailedAttempt(int failedAttempt) {  
 this.failedAttempt = failedAttempt;  
 }

public Date getLockTime() {  
 return lockTime;  
 }  
  
 public void setLockTime(Date lockTime) {  
 this.lockTime = lockTime;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getUserName() {  
 return username;  
 }  
  
 public void setUserName(String userName) {  
 this.username = userName;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public boolean isActive() {  
 return active;  
 }  
  
 public void setActive(boolean active) {  
 this.active = active;  
 }  
  
 public String getRoles() {  
 return roles;  
 }  
  
 public void setRoles(String roles) {  
 this.roles = roles;  
 }  
}

**UserRepository.java**

package SpringSecurity.Assignment04.repository;  
  
import java.util.Optional;  
  
import SpringSecurity.Assignment04.Model.User;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends JpaRepository<User,Integer> {  
 Optional<User> findByUsername(String username);  
  
}

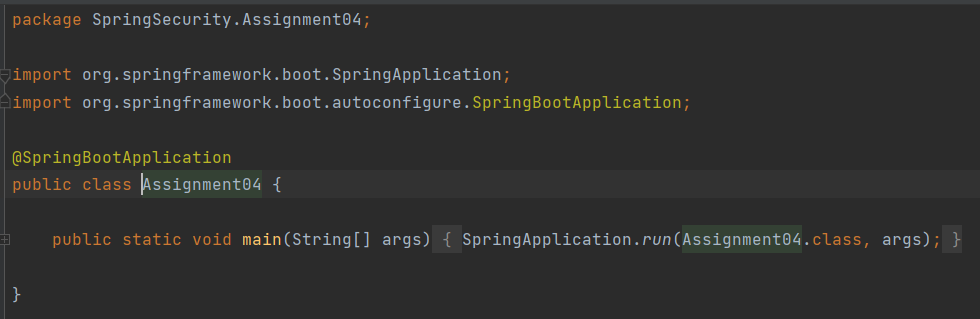
**MyUserDetailsService.java**

package SpringSecurity.Assignment04.services;  
  
import SpringSecurity.Assignment04.Model.MyUserDetails;  
import SpringSecurity.Assignment04.Model.User;  
import SpringSecurity.Assignment04.repository.UserRepository;  
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 java.util.Optional;  
  
@Service  
public class MyUserDetailsService implements UserDetailsService {  
  
@Autowired  
UserRepository userRepository;  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 Optional<User>user =userRepository.findByUsername(username);  
 user.orElseThrow(()-> new UsernameNotFoundException("Not found"));  
  
 return user.map(MyUserDetails::new).get();  
 }  
}

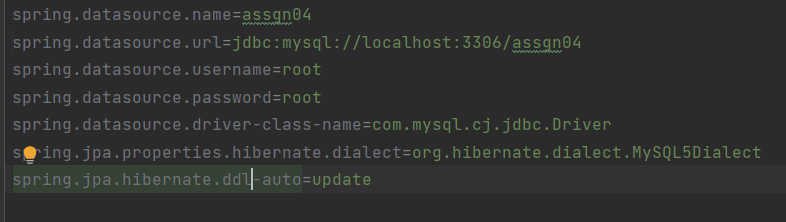
**UserService.java**

package SpringSecurity.Assignment04.services;  
  
import SpringSecurity.Assignment04.Model.User;  
import SpringSecurity.Assignment04.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.Date;  
  
@Service  
public class UserService {  
 @Autowired  
 UserRepository userRepository;  
  
 public User getUserByName(String username){  
 return userRepository.findByUsername(username).get();  
 }  
  
 public void increaseFailedAttempts(User user) {  
 int newFailedAttemp = user.getFailedAttempt()+1;  
 user.setFailedAttempt(newFailedAttemp);  
 userRepository.save(user);  
 }  
  
 public void lock(User user) {  
 user.setAccountNonLocked(false);  
 user.setLockTime(new Date());  
 userRepository.save(user);  
 }  
}

**Assignment04.java**

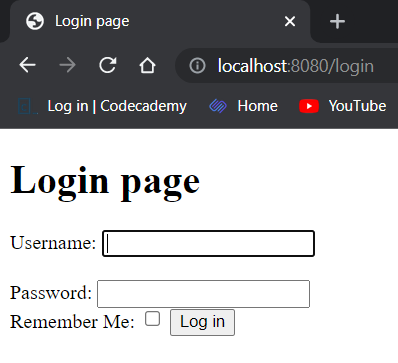


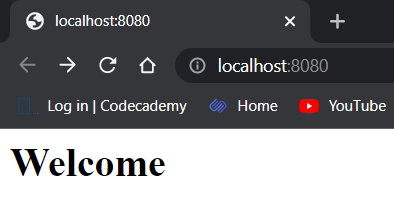
Application.Properites



Database User







5) Modify the above application to implement “remember me” functionality.

**MyConfig.java**

package SpringSecurity.Assignment05.config;  
  
import java.io.IOException;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import SpringSecurity.Assignment2.service.CustomUserDetailService;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.web.authentication.AuthenticationFailureHandler;

@Configuration  
@EnableWebSecurity  
public class MyConfig extends WebSecurityConfigurerAdapter {  
  
 static int *count* = 0;

@Autowired  
 private CustomUserDetailService customUserDetailService;  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
  
 http  
 .authorizeRequests()  
 .antMatchers("/signin").permitAll()  
 .anyRequest()  
 .authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/signin")  
 .loginProcessingUrl("/dologin")  
 .defaultSuccessUrl("/admin")  
 .failureHandler(new AuthenticationFailureHandler() {  
 @Override  
 public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,  
 AuthenticationException exception) throws IOException, ServletException {  
  
 System.*out*.println("A user has failed to login. Error: " + exception.getMessage());  
 if (*count* == 3) {  
  
 }  
 response.sendRedirect("signin?error");  
 }  
  
 })  
 .and()  
 .rememberMe().userDetailsService(customUserDetailService)  
 .tokenValiditySeconds(3 \* 24 \* 60 \* 60);  
 }  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth  
 .userDetailsService(customUserDetailService)  
 .passwordEncoder(passwordEncoder());  
 }

@Bean  
 public BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder(10);  
 }  
}

**HomeController.java**

package SpringSecurity.Assignment05.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.ResponseBody;  
  
@Controller  
public class HomeController {  
  
 @GetMapping("/signin")  
 public String login() {  
 return "login.html";  
 }  
  
 @GetMapping("/admin")  
 @ResponseBody  
 public String hellWorld() {  
 return "Hello World!!";  
 }  
  
 @PostMapping("login-fail")  
 public String handleFailedLogin() {  
 System.*out*.println("A User has failed to login");  
  
 return "redirect:/signin?error";  
 }  
}

**CustomUserDetail.java**

package SpringSecurity.Assignment05.model;  
  
import java.util.Collection;  
import java.util.HashSet;  
  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetail implements UserDetails {  
  
 private User user;  
  
 public CustomUserDetail(User user) {  
 this.user = user;  
 }  
  
 public CustomUserDetail() {  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
  
 HashSet<SimpleGrantedAuthority> set = new HashSet<>();  
 set.add(new SimpleGrantedAuthority(this.user.getRole()));  
  
 return set;  
 }  
  
 @Override  
 public String getPassword() {  
  
 return this.user.getPassword();  
 }  
  
 @Override  
 public String getUsername() {  
  
 return this.user.getUsername();  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
  
 return true;  
 }  
}

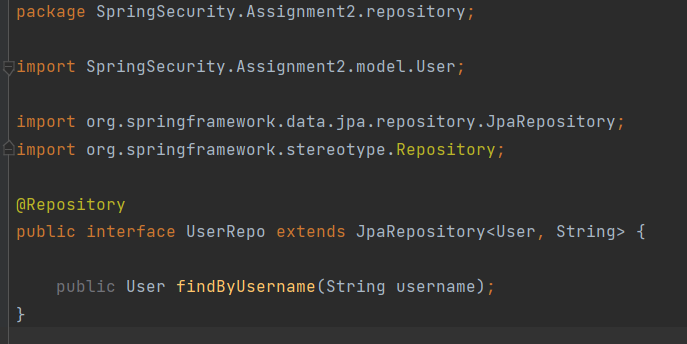
package SpringSecurity.Assignment05.model;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;

@Entity  
public class User {  
  
 @Id  
 String username;  
 String password;  
 String email;  
 String role;  
  
 public User() {  
 }  
  
 public User(String username, String password, String email, String role) {  
 this.username = username;  
 this.password = password;  
 this.email = email;  
 this.role = role;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getRole() {  
 return role;  
 }

public void setRole(String role) {  
 this.role = role;  
 }

@Override  
 public String toString() {  
 return "User [email=" + email + ", password=" + password + ", role=" + role + ", username=" + username + "]";  
 }  
  
}

**UserRepo.java**



**CustomUserDetailService.java**

package SpringSecurity.Assignment05.service;  
  
import SpringSecurity.Assignment05.model.CustomUserDetail;  
import SpringSecurity.Assignment05.model.User;  
import SpringSecurity.Assignment05.repository.UserRepo;  
  
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;

@Service  
public class CustomUserDetailService implements UserDetailsService {  
  
 @Autowired  
 private UserRepo userRepo;

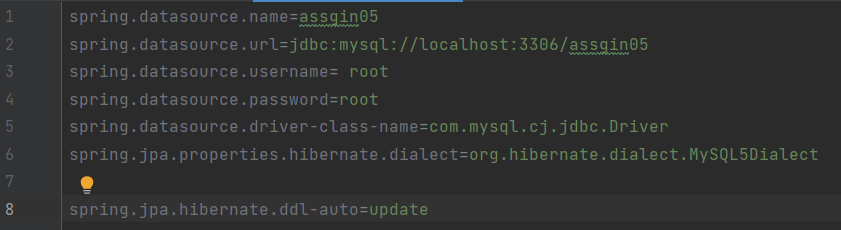
@Override  
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
  
 User user = this.userRepo.findByUsername(username);  
  
 if (user == null) {  
 throw new UsernameNotFoundException("INVALID USERNAME!!");  
 }  
  
 return new CustomUserDetail(user);  
 }  
  
}

**Assignment05.java**

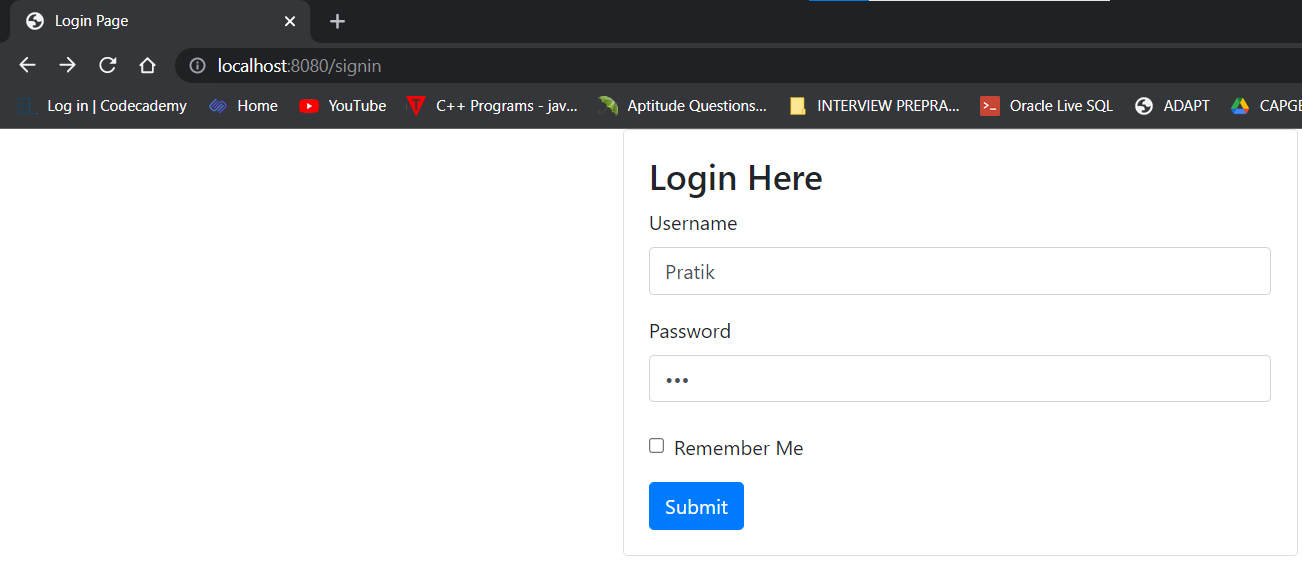
package SpringSecurity.Assignment05;  
  
import SpringSecurity.Assignment05.repository.UserRepo;  
import SpringSecurity.Assignment05.model.User;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
@SpringBootApplication  
public class Assignment05 implements CommandLineRunner {  
  
 @Autowired  
 private UserRepo userRepo;  
  
 @Autowired  
 private BCryptPasswordEncoder bCryptPasswordEncoder;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Assignment05.class, args);  
 }

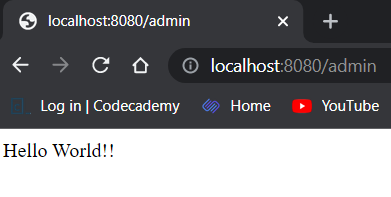
@Override  
 public void run(String... args) throws Exception {  
  
 User user1 = new User();  
 user1.setEmail("pratik@gmail.com");  
 user1.setUsername("Pratik");  
 user1.setPassword(bCryptPasswordEncoder.encode("abc"));  
 user1.setRole("ROLE\_ADMIN");  
  
 userRepo.save(user1);  
  
 User user2 = new User();  
 user2.setEmail("virat@gmail.com");  
 user2.setUsername("Virat");  
 user2.setPassword(bCryptPasswordEncoder.encode("cricket"));  
 user2.setRole("ROLE\_NORMAL");  
  
 userRepo.save(user2);  
 }  
  
}

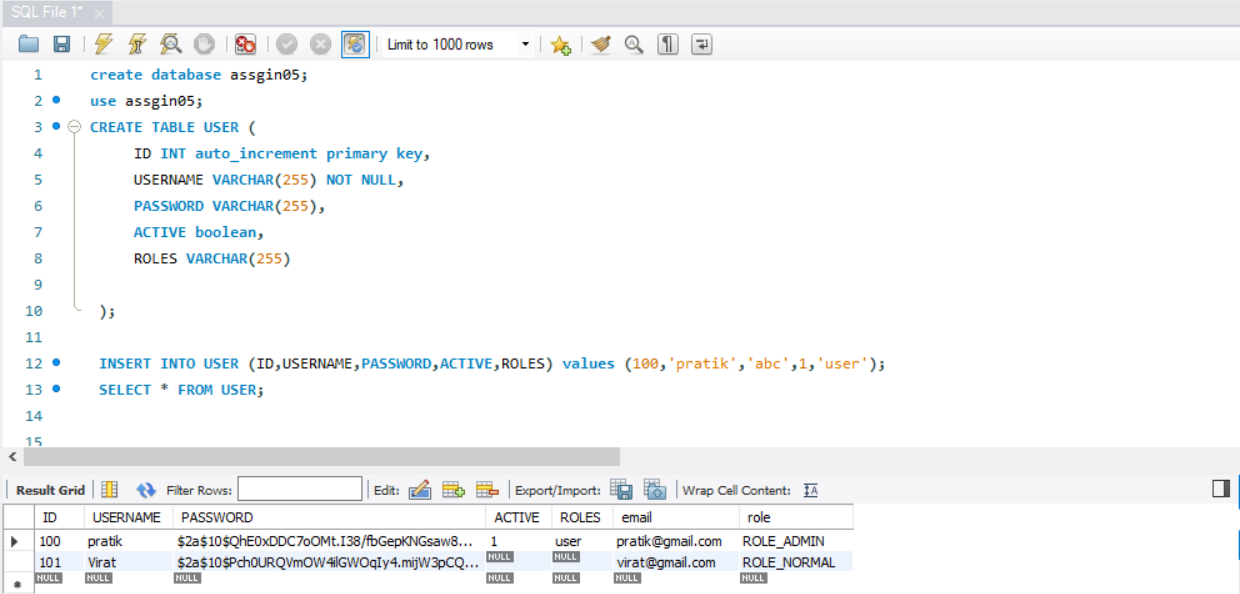
**Application.Properties**

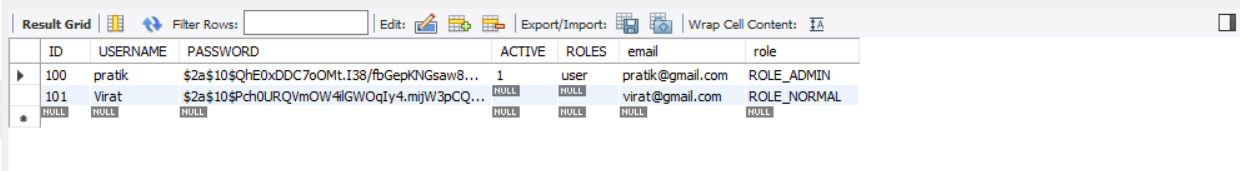


**Output :-**









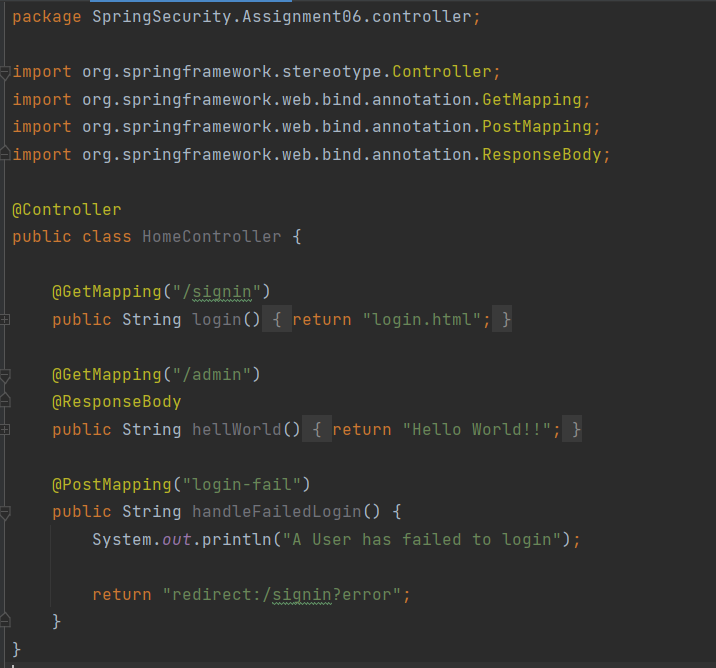
6) Modify the above application to secure the password by encoding it. (You may use some encryption technique)

**MyConfig.java**

package SpringSecurity.Assignment06.config;  
  
import java.io.IOException;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import SpringSecurity.Assignment06.service.CustomUserDetailService;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.web.authentication.AuthenticationFailureHandler;  
  
@Configuration  
@EnableWebSecurity  
public class MyConfig extends WebSecurityConfigurerAdapter {  
  
 static int *count* = 0;  
 @Autowired  
 private CustomUserDetailService customUserDetailService;

@Override  
 protected void configure(HttpSecurity http) throws Exception {  
  
 http  
 .authorizeRequests()  
 .antMatchers("/signin").permitAll()  
 .anyRequest()  
 .authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/signin")  
 .loginProcessingUrl("/dologin")  
 .defaultSuccessUrl("/admin")  
 .failureHandler(new AuthenticationFailureHandler() {  
 @Override  
 public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,  
 AuthenticationException exception) throws IOException, ServletException {  
  
 System.*out*.println("A user has failed to login. Error: " + exception.getMessage());  
 if (*count* == 3) {  
  
 }  
 response.sendRedirect("signin?error");  
 }  
  
 })  
 .and()  
 .rememberMe().userDetailsService(customUserDetailService)  
 .tokenValiditySeconds(3 \* 24 \* 60 \* 60);  
 }  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth  
 .userDetailsService(customUserDetailService)  
 .passwordEncoder(passwordEncoder());  
 }  
  
 @Bean  
 public BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder(10);  
 }  
}

**HomeController.java**



**CustomUserDetail.java**

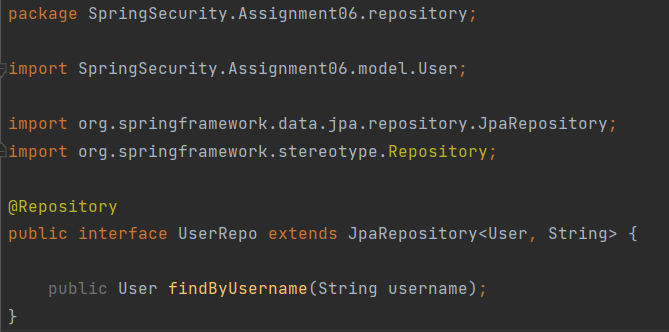
package SpringSecurity.Assignment06.model;  
  
import java.util.Collection;  
import java.util.HashSet;  
  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetail implements UserDetails {  
  
 private User user;  
  
 public CustomUserDetail(User user) {  
 this.user = user;  
 }

public CustomUserDetail() {  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
  
 HashSet<SimpleGrantedAuthority> set = new HashSet<>();  
 set.add(new SimpleGrantedAuthority(this.user.getRole()));  
  
 return set;  
 }  
  
 @Override  
 public String getPassword() {  
  
 return this.user.getPassword();  
 }  
  
 @Override  
 public String getUsername() {  
  
 return this.user.getUsername();  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
  
 return true;  
 }  
  
}

package SpringSecurity.Assignment06.model;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
  
@Entity  
public class User {  
  
 @Id  
 String username;  
 String password;  
 String email;  
 String role;  
  
 public User() {  
 }  
  
 public User(String username, String password, String email, String role) {  
 this.username = username;  
 this.password = password;  
 this.email = email;  
 this.role = role;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 @Override  
 public String toString() {  
 return "User [email=" + email + ", password=" + password + ", role=" + role + ", username=" + username + "]";  
 }  
  
}

**UserRepo**



package SpringSecurity.Assignment06.service;  
  
import SpringSecurity.Assignment06.repository.UserRepo;  
import SpringSecurity.Assignment06.model.CustomUserDetail;  
import SpringSecurity.Assignment06.model.User;  
  
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;  
  
@Service  
public class CustomUserDetailService implements UserDetailsService {  
  
 @Autowired  
 private UserRepo userRepo;  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
  
 User user = this.userRepo.findByUsername(username);  
  
 if (user == null) {  
 throw new UsernameNotFoundException("INVALID USERNAME!!");  
 }  
  
 return new CustomUserDetail(user);  
 }  
  
}

**Assignment06.java**

package SpringSecurity.Assignment06;  
  
import SpringSecurity.Assignment06.model.User;  
import SpringSecurity.Assignment06.repository.UserRepo;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
@SpringBootApplication  
public class Assignment06 implements CommandLineRunner {  
  
 @Autowired  
 private UserRepo userRepo;  
  
 @Autowired  
 private BCryptPasswordEncoder bCryptPasswordEncoder;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Assignment06.class, args);  
 }  
  
 @Override  
 public void run(String... args) throws Exception {  
  
 User user1 = new User();  
 user1.setEmail("pratik@gmail.com");  
 user1.setUsername("Pratik");  
 user1.setPassword(bCryptPasswordEncoder.encode("abc"));  
 user1.setRole("ROLE\_ADMIN");  
  
 userRepo.save(user1);  
  
 User user2 = new User();  
 user2.setEmail("virat@gmail.com");  
 user2.setUsername("virat");  
 user2.setPassword(bCryptPasswordEncoder.encode("cricket"));  
 user2.setRole("ROLE\_NORMAL");  
  
 userRepo.save(user2);  
 }  
  
}

OUTPUT :

