public class CustomUserDetails implements UserDetails{  
  
 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();  
 }  
   
   
   
}

@Component  
public class CustomUserDetailsService implements UserDetailsService{  
  
 UserRepo userRepo;  
 public CustomUserDetailsService(UserRepo userRepo) {  
 this.userRepo = userRepo;  
 }  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
   
 User user = userRepo.findByUserName(username);  
 if(user==null)  
 { System.*out*.println("User not found");  
 throw new UsernameNotFoundException("User not found");  
 }  
 return new CustomUserDetails(user);  
 }  
   
}

@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 UserDetailsService userDetailsService;  
 public SecurityConfig(UserDetailsService userDetailsService) {  
 this.userDetailsService = userDetailsService;  
 }  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception  
 {  
 httpSecurity  
 .csrf(csrf->csrf.disable())  
   
 .authorizeHttpRequests(request->  
 request.requestMatchers("adduser","login").permitAll()  
 .anyRequest().authenticated())  
   
 .httpBasic(Customizer.*withDefaults*());  
 return httpSecurity.build();  
  
 }  
  
  
  
 @Bean  
 public AuthenticationProvider authenticationProvider()  
 {  
 DaoAuthenticationProvider provider=new DaoAuthenticationProvider();  
 // provider.setUserDetailsService(userDetailsService());  
   
 provider.setUserDetailsService((UserDetailsService) userDetailsService);  
 // provider.setPasswordEncoder(NoOpPasswordEncoder.getInstance());  
 provider.setPasswordEncoder(bCryptPasswordEncoder());  
 return provider;  
  
 }  
  
 @Bean  
 public BCryptPasswordEncoder bCryptPasswordEncoder()  
 {  
 return new BCryptPasswordEncoder(10);  
 }  
}

@Service  
public class UserPassService {  
   
 // @Autowired  
 private final UserRepo userRepo;  
 // @Autowired  
 private final BCryptPasswordEncoder bCryptPasswordEncoder;  
  
 public UserPassService(UserRepo userRepo, BCryptPasswordEncoder bCryptPasswordEncoder) {  
 this.userRepo = userRepo;  
 this.bCryptPasswordEncoder = bCryptPasswordEncoder;  
 }  
  
 public User createUser(User user) {  
 user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));  
 return userRepo.save(user);  
 }  
}