1.Add following 3 dependencies

```
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
```

---

2. Add the JWTUtility class in util package

```
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import java.io.Serializable;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Component
public class JWTUtility implements Serializable {

    private static final long serialVersionUID = 234234523523L;

    public static final long JWT_TOKEN_VALIDITY = 24 * 60 * 60; // seconds

    @Value("${jwt.secret}")
    private String secretKey;

    //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 retrieving any information from token we will need the secret key
    private Claims getAllClaimsFromToken(String token) {
        return Jwts.parser().setSigningKey(secretKey).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.
    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, secretKey).compact();
    }


    //validate token
    public Boolean validateToken(String token, UserDetails userDetails) {
        final String username = getUsernameFromToken(token);
        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
```

```
    }
}
```

---

3. Add the JwtRequest and JwtResponse entity in entities package

```
public class JwtRequest
{

      private String username;
      private String password;

      public JwtRequest() {
          super();
          // TODO Auto-generated constructor stub
     }

     public JwtRequest(String username, String password) {
          super();
          this.username = username;
          this.password = password;
     }

     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;
     }
}
```

JwtResponse.java

```
public class JwtResponse
{
      private String token;
      private String createdAt;
```

```
     private String expiresOn;

    public JwtResponse() {
         super();
         // TODO Auto-generated constructor stub
    }

    public JwtResponse(String token) {
         super();
         this.token = token;
    }



    public JwtResponse(String token, String createdAt, String expiresOn) {
         super();
         this.token = token;
         this.createdAt = createdAt;
         this.expiresOn = expiresOn;
    }

    public String getJwtToken() {
         return token;
    }

    public void setJwtToken(String token) {
         this.token = token;
    }

    public String getCreatedAt() {
         return createdAt;
    }

    public void setCreatedAt(String createdAt) {
         this.createdAt = createdAt;
    }

    public String getExpiresOn() {
         return expiresOn;
    }

    public void setExpiresOn(String expiresOn) {
         this.expiresOn = expiresOn;
    }

}
```

---

4. Make the following entry in application.properties

```
jwt.secret=secretkey123
```

---

5. Add the service in services package

```
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.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.ArrayList;

@Service
public class UserService implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String userName) throws
UsernameNotFoundException {

        //Logic to get the user form the Database

        return new User("leenaai","tivE%crea4",new ArrayList<>());
    }
}
```

---

6. Add a filter under a jwt package

```
import java.io.IOException;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import ae.cocacola.datagateway.jwt.service.UserService;
import ae.cocacola.datagateway.jwt.utility.JWTUtility;

@Component
public class JwtFilter extends OncePerRequestFilter {

    @Autowired
    private JWTUtility jwtUtility;

    @Autowired
    private UserService userService;

    @Override
    protected void doFilterInternal(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, FilterChain filterChain) throws
ServletException, IOException {
        String authorization = httpServletRequest.getHeader("Authorization");
        String token = null;
        String userName = null;

        if(null != authorization && authorization.startsWith("Bearer ")) {
            token = authorization.substring(7);
            userName = jwtUtility.getUsernameFromToken(token);
        }

        if(null != userName && SecurityContextHolder.getContext().getAuthentication()
== null) {
            UserDetails userDetails
                    = userService.loadUserByUsername(userName);

            if(jwtUtility.validateToken(token,userDetails)) {
                UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken
                        = new UsernamePasswordAuthenticationToken(userDetails,
                        null, userDetails.getAuthorities());

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


SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationTok
en);
            }
```

```
        }
        filterChain.doFilter(httpServletRequest, httpServletResponse);
    }
}
```

---

7. Add a config in main package

```
package com.scb.omega;

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.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
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.WebSecurityConfigurer
Adapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.scb.omega.jwt.JwtFilter;
import com.scb.omega.service.impl.UserService;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter{

    @Autowired
    private UserService userService;

    @Autowired
    private JwtFilter jwtFilter;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

        auth.userDetailsService(userService);
    }
```

```
    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {

     http.csrf()
        .disable()
        .authorizeRequests()
        .antMatchers("/authenticate")
        .permitAll()
        .antMatchers("/chatbot/**")
        .authenticated()
        .and()
        .sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

    }
}
```

---

8. In main class add the following method

```
package com.scb.omega;

import javax.annotation.Resource;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

import com.scb.omega.service.impl.FileStorageService;

@SpringBootApplication
public class SpringdbplaygroundApplication implements CommandLineRunner {

     @Resource
     FileStorageService storageService;

     public static void main(String[] args) {
```

```
		SpringApplication.run(SpringdbplaygroundApplication.class, args);
	}

	@Override
	public void run(String... args) throws Exception {
		//storageService.init();

	}


	@Bean
	public PasswordEncoder passwordEncoder() {
		return NoOpPasswordEncoder.getInstance();
	}


}
```

---

9. Add the controller

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

import org.springframework.beans.factory.annotation.Autowired;
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.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.scb.omega.entities.vo.JwtRequest;
import com.scb.omega.entities.vo.JwtResponse;
import com.scb.omega.service.impl.UserService;
import com.scb.omega.utils.JWTUtility;

@RestController
public class LoginController {

    @Autowired
    private JWTUtility jwtUtility;

    @Autowired
    private AuthenticationManager authenticationManager;
```

```
    @Autowired
    private UserService userService;

    @GetMapping("/jwt")
    public String home() {
        return "Welcome to SCB!";
    }

    @PostMapping("/authenticate")
    public JwtResponse authenticate(@RequestBody JwtRequest jwtRequest) throws
Exception{

        try {
            authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(
                            jwtRequest.getUsername(),
                            jwtRequest.getPassword()
                    )
            );
        } catch (BadCredentialsException e) {
            throw new Exception("INVALID_CREDENTIALS", e);
        }

        final UserDetails userDetails
                = userService.loadUserByUsername(jwtRequest.getUsername());

        String token =
                jwtUtility.generateToken(userDetails);
        LocalDateTime now=LocalDateTime.now();
        DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        String createdAt = now.format(format);
        LocalDateTime expire=now.plusHours(24);
        String expiresOn=expire.format(format);

        token = "Bearer " + token;

        return  new JwtResponse(token,createdAt,expiresOn);
    }
}
```