1. Add Dependencies

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.11.5</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
</dependencies>
```

2. `JwtTokenUtil` Class

```java
package com.example.security.config;

import io.jsonwebtoken.*;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class JwtTokenUtil {

    private final String SECRET_KEY = "your-secret-key";

    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + 864(
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
                .compact();
    }

    public boolean validateToken(String token) {
        try {
            Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token)
            return true;
        } catch (JwtException e) {
            return false;
        }
    }

    public String getUsernameFromToken(String token) {
        return Jwts.parser()
                .setSigningKey(SECRET_KEY)
                .parseClaimsJws(token)
                .getBody()
                .getSubject();
    }
}
```

3. `JwtAuthenticationFilter` Class

```
package com.example.security.filters;

import com.example.security.config.JwtTokenUtil;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtTokenUtil jwtTokenUtil;

    public JwtAuthenticationFilter(JwtTokenUtil jwtTokenUtil) {
        this.jwtTokenUtil = jwtTokenUtil;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request, FilterCha
        String token = request.getHeader("Authorization");

        if (token != null && token.startsWith("Bearer ")) {
            token = token.substring(7);

            if (jwtTokenUtil.validateToken(token)) {
                String username = jwtTokenUtil.getUsernameFromToken(toker
                SecurityContextHolder.getContext().setAuthentication(new
            }
        }

        filterChain.doFilter(request, null);
    }
}
```

4. CustomUserDetailsService Class

```
package com.example.security.service;

import com.example.security.model.User;
import com.example.security.repository.UserRepository;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundExce
import org.springframework.stereotype.Service;

@Service
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws Usernam
        User user = userRepository.findByUsername(username)
```

```
                .orElseThrow(() -> new UsernameNotFoundException("User nc
        return new org.springframework.security.core.userdetails.User(use
    }
}
```

5. Spring Security Configuration (`SecurityConfig`)

```
package com.example.security.config;

import com.example.security.filters.JwtAuthenticationFilter;
import com.example.security.service.CustomUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.buil
import org.springframework.security.config.annotation.web.builders.HttpSe
import org.springframework.security.config.annotation.web.configuration.E
import org.springframework.security.config.annotation.web.configuration.W
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAu

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private final CustomUserDetailsService userDetailsService;
    private final JwtTokenUtil jwtTokenUtil;

    public SecurityConfig(CustomUserDetailsService userDetailsService, Jw
        this.userDetailsService = userDetailsService;
        this.jwtTokenUtil = jwtTokenUtil;
    }

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

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeRequests()
            .antMatchers("/auth/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .addFilterBefore(new JwtAuthenticationFilter(jwtTokenUtil), U
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Ex
        auth.userDetailsService(userDetailsService).passwordEncoder(passw
    }
}
```

6. Authentication Controller (`AuthController`)

```
package com.example.security.controllers;

import com.example.security.config.JwtTokenUtil;
import com.example.security.service.CustomUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/auth")
public class AuthController {

    private final JwtTokenUtil jwtTokenUtil;
    private final CustomUserDetailsService userDetailsService;

    @Autowired
    public AuthController(JwtTokenUtil jwtTokenUtil, CustomUserDetailsSer
        this.jwtTokenUtil = jwtTokenUtil;
        this.userDetailsService = userDetailsService;
    }

    @PostMapping("/login")
    public String login(@RequestParam String username, @RequestParam Stri
        if ("user".equals(username) && "password".equals(password)) {
            return jwtTokenUtil.generateToken(username);
        }
        return "Invalid Credentials";
    }
}
```

7. User Model and Repository

```java
package com.example.security.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class User {
    @Id
    private Long id;
    private String username;
    private String password;

    // Getters and Setters
}
```

```java
package com.example.security.repository;

import com.example.security.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
}
```

This configuration uses **AuthenticationManagerBuilder** to set up **user authentication** with JWT, ensuring that every request to secured endpoints passes through the **JWT filter** to authenticate the token. The `CustomUserDetailsService` is used to load user details from the database and authenticate based on the provided **username** and **password**.