```java
package com.wecp.library.controller;

import com.wecp.library.controller.exception.UserNotSubscribedException;
import com.wecp.library.domain.Issue;
import com.wecp.library.domain.User;
import com.wecp.library.repository.IssueRepository;
import com.wecp.library.repository.UserRepository;

import org.apache.catalina.connector.Response;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

/**
 * REST controller for managing library system process
 */
@RestController
@RequestMapping("/api/v1")
public class LibraryController {

    @Autowired
    private IssueRepository issueRepository;

    @Autowired
    private UserRepository userRepository;

    /**
     * {@code POST  /issueBook} : Create a new issue.
     *
     * @param issue the issue to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     *         the user, or if does not exist, return with status "noContent".
     *         If user is not subscribed, throw {@link
UserNotSubscribedException}
     */
    @PostMapping("/issue-book")
    public ResponseEntity<Issue> issueBook(@RequestBody Issue issue) {
        Optional<User> user = userRepository.findById(issue.getUser().getId());
        if (!user.isPresent()) {
            return ResponseEntity.noContent().build();
        }
        if (!user.get().getSubscribed()) {
```

```
            throw new UserNotSubscribedException("User not subscribed");
        }

        return ResponseEntity.ok().body(issueRepository.save(issue));
    }

    /**
     * {@code POST  /user} : Create a new user.
     *
     * @param user the user to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     *         the new user
     */
    @PostMapping("/user")
    public ResponseEntity<User> createUser(@RequestBody User user) {
        User userData = userRepository.save(user);
        return ResponseEntity.ok().body(userData);
    }

    /**
     * {@code GET  /renew-user-subscription/:id} : Send userId, set user
     * subscription to true
     *
     * @param id the id of the user to renew subscription.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     *         the user, or if does not exist, return with status "noContent".
     */
    @GetMapping("renew-user-subscription/{id}")
    public ResponseEntity<User> renewUserSubscription(@PathVariable Long id) {
        Optional<User> user = userRepository.findById(id);
        if(user.isEmpty()){
            return ResponseEntity.noContent().build();
        }
        user.get().setSubscribed(true);
        return ResponseEntity.ok().body(userRepository.save(user.get()));

    }
}
```

```java
package com.wecp.library.security;

import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpStatus;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authentica
tionManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
y;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfi
gurerAdapter;
import org.springframework.security.web.authentication.HttpStatusEntryPoint;

/**
 * Configure Spring Security class here. Don't forget to extend the class with
 * the necessary Spring Security class.
 * user and renew-user-subscription APIs must be authenticated and issue-book
 * must be permitAll.
 */
@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        // TODO Auto-generated method stub
        auth.authenticationProvider(new DaoAuthenticationProvider());
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        // TODO Auto-generated method stub
        http.authorizeRequests().antMatchers("/api/v1/renew-user-
subscription/**").permitAll().anyRequest()
                .authenticated()
                .and().exceptionHandling().authenticationEntryPoint(new
HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED));
    }
```

```
}
```