

پروژه دوره Backend with Spring Boot Java

شرکت داتین

اعضای گروه: پریناز عاکف، مریم مهدی زاده، زینب تیموری

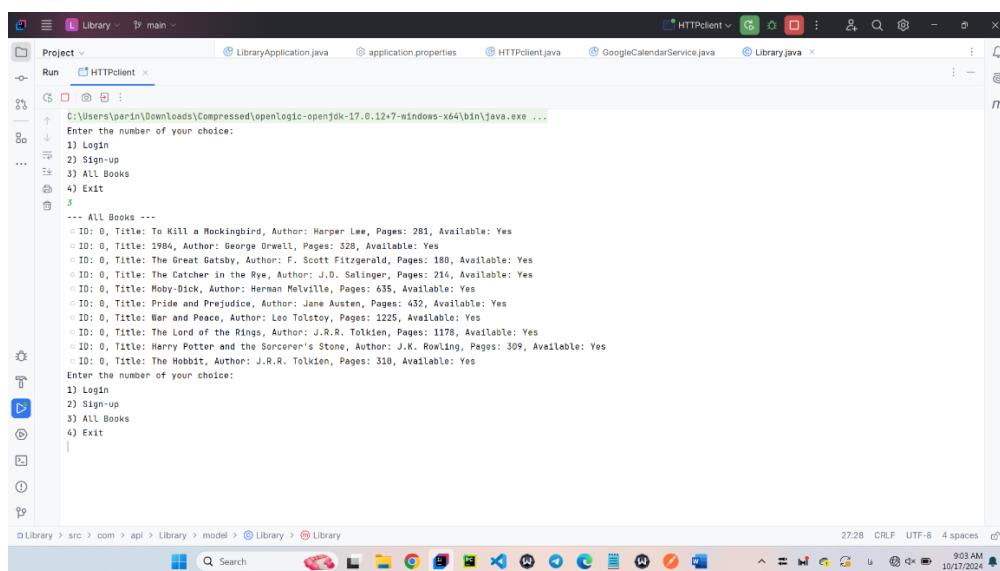
توضیح پروژه

برنامه کتابخانه برای مدیریت رزرو کتاب توسط کاربران

مرحله اول

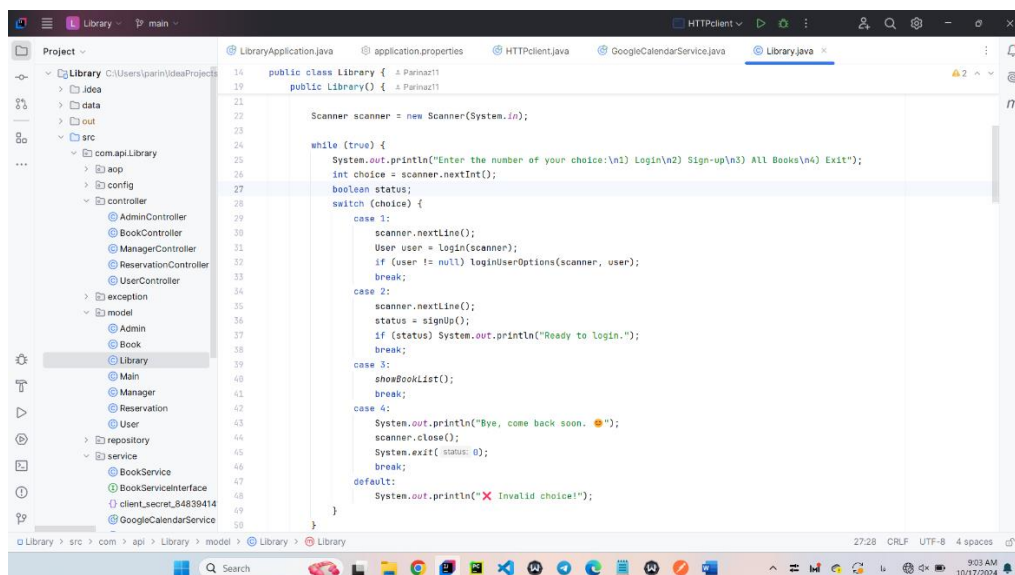
تشکیل پروژه کتابخانه، به صورتی که کاربر امکان تعامل با ترمینال داشته باشد و داده ها در ArrayList ذخیره شوند.

تعامل کاربر با کتابخانه:



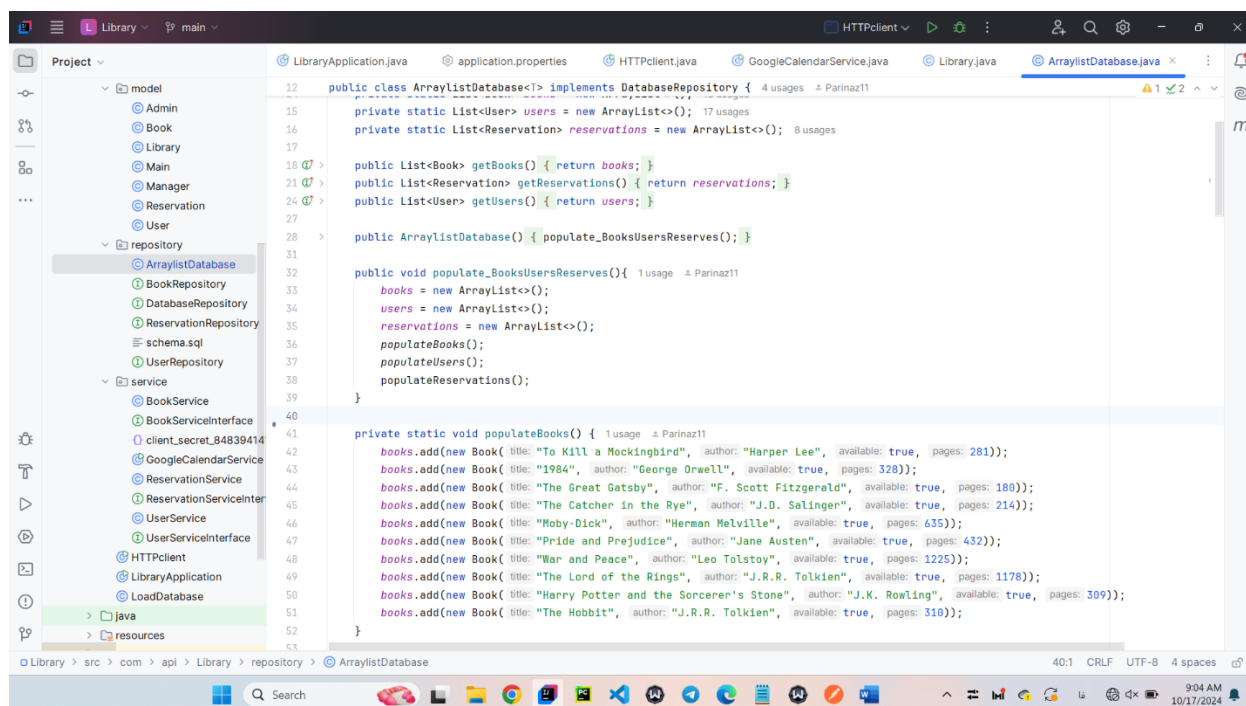
```
Project Library
Run HTTPClient
C:\Users\parin\Downloads\Compressed\openlogic-openjdk-17.0.12+7-windows-x64\bin\java.exe ...
Enter the number of your choice:
1) Login
2) Sign-up
3) All Books
4) Exit
3
--- All Books ---
ID: 0, Title: To Kill a Mockingbird, Author: Harper Lee, Pages: 281, Available: Yes
ID: 0, Title: 1984, Author: George Orwell, Pages: 328, Available: Yes
ID: 0, Title: The Great Gatsby, Author: F. Scott Fitzgerald, Pages: 180, Available: Yes
ID: 0, Title: The Catcher in the Rye, Author: J.D. Salinger, Pages: 214, Available: Yes
ID: 0, Title: Moby-Dick, Author: Herman Melville, Pages: 435, Available: Yes
ID: 0, Title: Pride and Prejudice, Author: Jane Austen, Pages: 432, Available: Yes
ID: 0, Title: War and Peace, Author: Leo Tolstoy, Pages: 1225, Available: Yes
ID: 0, Title: The Lord of the Rings, Author: J.R.R. Tolkien, Pages: 1178, Available: Yes
ID: 0, Title: Harry Potter and the Sorcerer's Stone, Author: J.K. Rowling, Pages: 309, Available: Yes
ID: 0, Title: The Hobbit, Author: J.R.R. Tolkien, Pages: 310, Available: Yes
Enter the number of your choice:
1) Login
2) Sign-up
3) All Books
4) Exit
|
```

نمونه ای از کد کلاس Library:



```
LibraryApplication.java application.properties HTTPClient.java GoogleCalendarService.java Library.java
14 public class Library {
15     public Library() {
16         Scanner scanner = new Scanner(System.in);
17
18         while (true) {
19             System.out.println("Enter the number of your choice:\n1) Login\n2) Sign-up\n3) All Books\n4) Exit*");
20             int choice = scanner.nextInt();
21             boolean status;
22             switch (choice) {
23                 case 1:
24                     scanner.nextLine();
25                     User user = login(scanner);
26                     if (user != null) loginUserOptions(scanner, user);
27                     break;
28                 case 2:
29                     scanner.nextLine();
30                     status = signUp();
31                     if (status) System.out.println("Ready to login.");
32                     break;
33                 case 3:
34                     showBookList();
35                     break;
36                 case 4:
37                     System.out.println("Bye, come back soon. 🍀");
38                     scanner.close();
39                     System.exit(0);
40                     break;
41                 default:
42                     System.out.println("Invalid choice!");
43             }
44         }
45     }
46 }
47
48
49
50
```

نمونه ای از کد ArrayList ها:



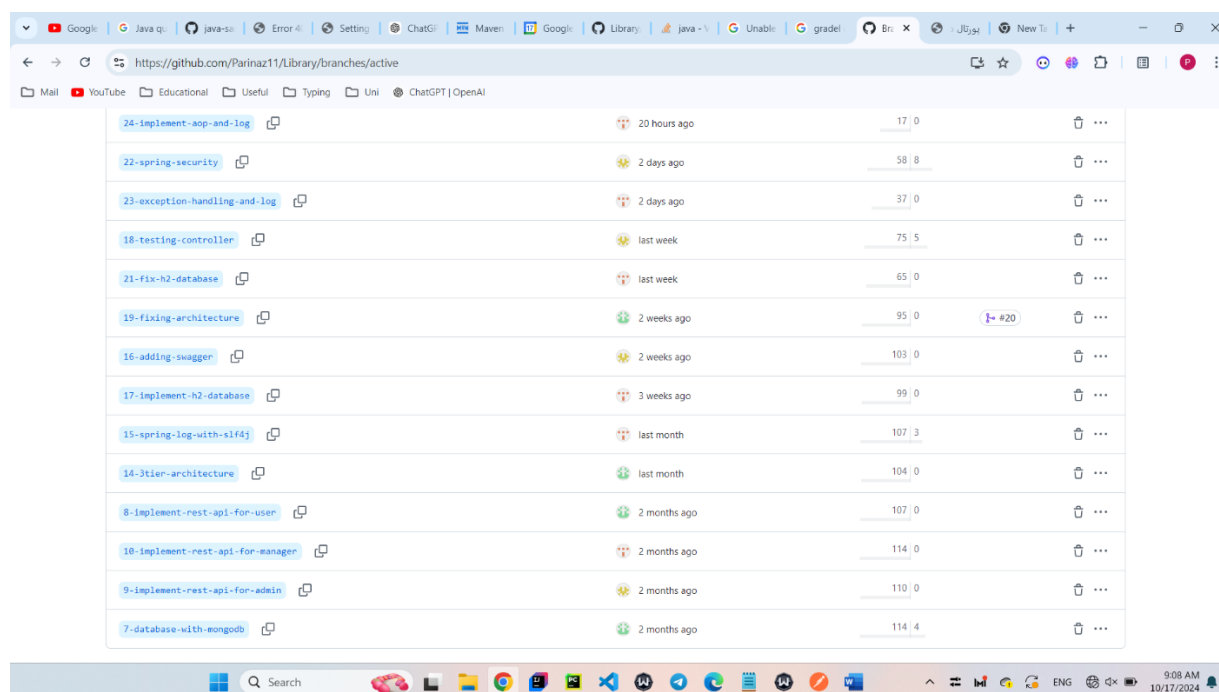
```
12 public class ArrayListDatabase<T> implements DatabaseRepository {
15     private static List<User> users = new ArrayList<>();
16     private static List<Reservation> reservations = new ArrayList<>();
17
18     public List<Book> getBooks() { return books; }
21     public List<Reservation> getReservations() { return reservations; }
24     public List<User> getUsers() { return users; }
27
28     public ArrayListDatabase() { populate_BooksUsersReserves(); }
31
32     public void populate_BooksUsersReserves(){
33         books = new ArrayList<>();
34         users = new ArrayList<>();
35         reservations = new ArrayList<>();
36         populateBooks();
37         populateUsers();
38         populateReservations();
39     }
40
41     private static void populateBooks() {
42         books.add(new Book( title: "To Kill a Mockingbird", author: "Harper Lee", available: true, pages: 281));
43         books.add(new Book( title: "1984", author: "George Orwell", available: true, pages: 328));
44         books.add(new Book( title: "The Great Gatsby", author: "F. Scott Fitzgerald", available: true, pages: 180));
45         books.add(new Book( title: "The Catcher in the Rye", author: "J.D. Salinger", available: true, pages: 214));
46         books.add(new Book( title: "Moby-Dick", author: "Herman Melville", available: true, pages: 635));
47         books.add(new Book( title: "Pride and Prejudice", author: "Jane Austen", available: true, pages: 432));
48         books.add(new Book( title: "War and Peace", author: "Leo Tolstoy", available: true, pages: 1225));
49         books.add(new Book( title: "The Lord of the Rings", author: "J.R.R. Tolkien", available: true, pages: 1178));
50         books.add(new Book( title: "Harry Potter and the Sorcerer's Stone", author: "J.K. Rowling", available: true, pages: 309));
51         books.add(new Book( title: "The Hobbit", author: "J.R.R. Tolkien", available: true, pages: 310));
52     }
53 }
```

مرحله دوم

تشکیل پروژه در گیت هاب و درست کردن Issue برای تشکیل branch و مدیریت تسک ها

لینک پروژه در گیت: <https://github.com/Parinaz11/Library.git>

نمونه ای از branch ها:



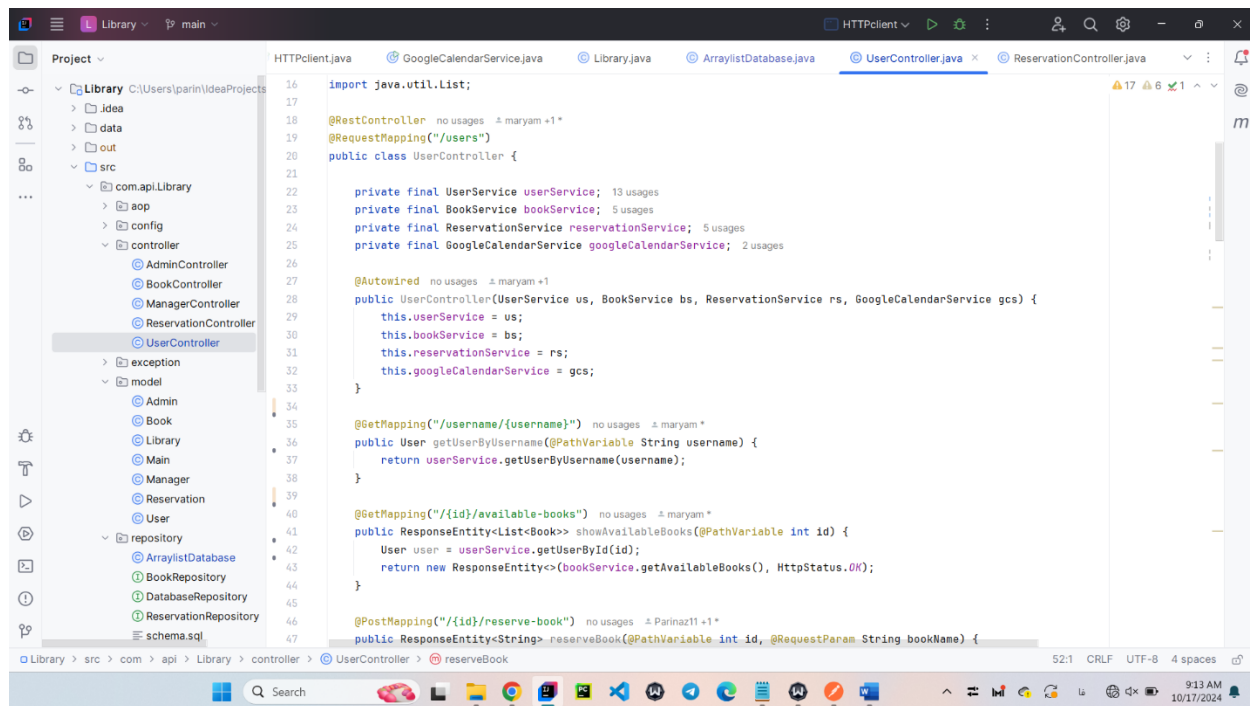
Branch Name	Last Commit	Commits
24-implement-aop-and-log	20 hours ago	17
22-spring-security	2 days ago	58
23-exception-handling-and-log	2 days ago	37
18-testing-controller	last week	75
21-fix-h2-database	last week	65
19-fixing-architecture	2 weeks ago	95
16-adding-swagger	2 weeks ago	103
17-implement-h2-database	3 weeks ago	99
15-spring-log-with-slf4j	last month	107
14-3tier-architecture	last month	104
8-implement-rest-api-for-user	2 months ago	107
10-implement-rest-api-for-manager	2 months ago	114
9-implement-rest-api-for-admin	2 months ago	110
7-database-with-mongodb	2 months ago	114

همکاری تیمی:

تعامل در گروه تلگرامی با منتور، تشکیل جلسات در Google Meet برای تعامل و رفع اشکالات

مرحله سوم

پیاده سازی Rest API و کلاس های Service و Controller



```
import java.util.List;

@RestController
@RequestMapping("/users")
public class UserController {

    private final UserService userService;
    private final BookService bookService;
    private final ReservationService reservationService;
    private final GoogleCalendarService googleCalendarService;

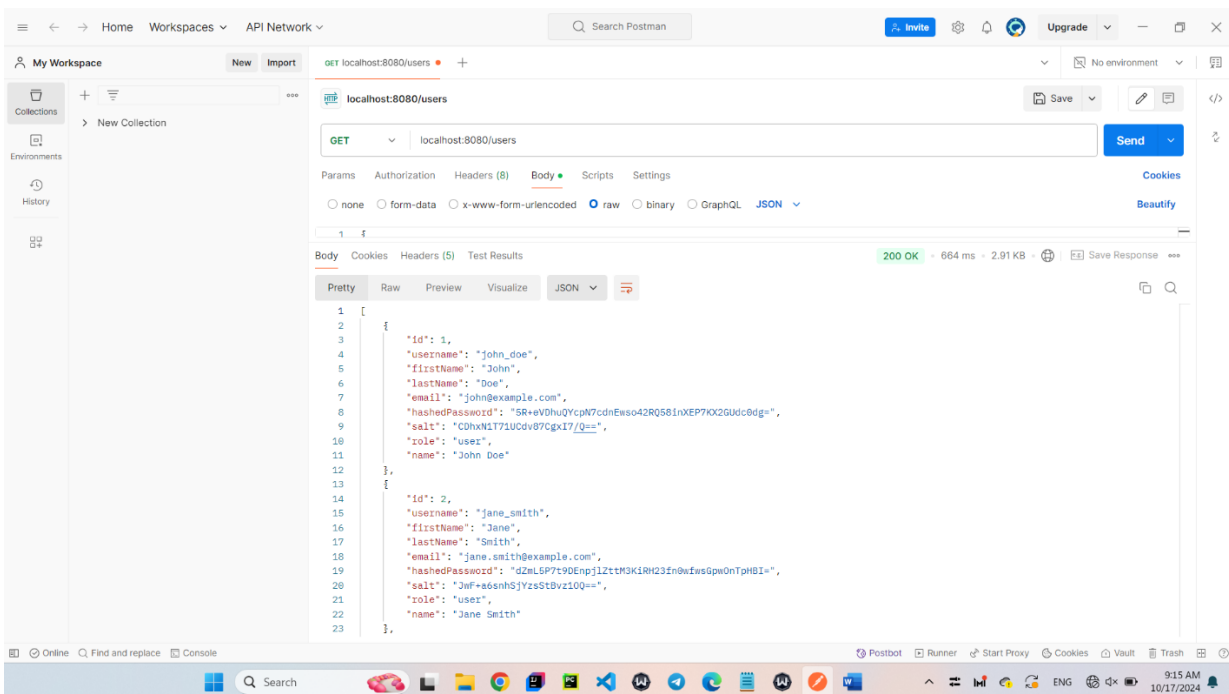
    @Autowired
    public UserController(UserService us, BookService bs, ReservationService rs, GoogleCalendarService gcs) {
        this.userService = us;
        this.bookService = bs;
        this.reservationService = rs;
        this.googleCalendarService = gcs;
    }

    @GetMapping("/username/{username}")
    public User getUserByUsername(@PathVariable String username) {
        return userService.getUserByUsername(username);
    }

    @GetMapping("/{id}/available-books")
    public ResponseEntity<List<Book>> showAvailableBooks(@PathVariable int id) {
        User user = userService.getUserById(id);
        return new ResponseEntity<>(bookService.getAvailableBooks(), HttpStatus.OK);
    }

    @PostMapping("/{id}/reserve-book")
    public ResponseEntity<String> reserveBook(@PathVariable int id, @RequestParam String bookName) {
```

تست نتیجه پیاده سازی Rest API با Postman:



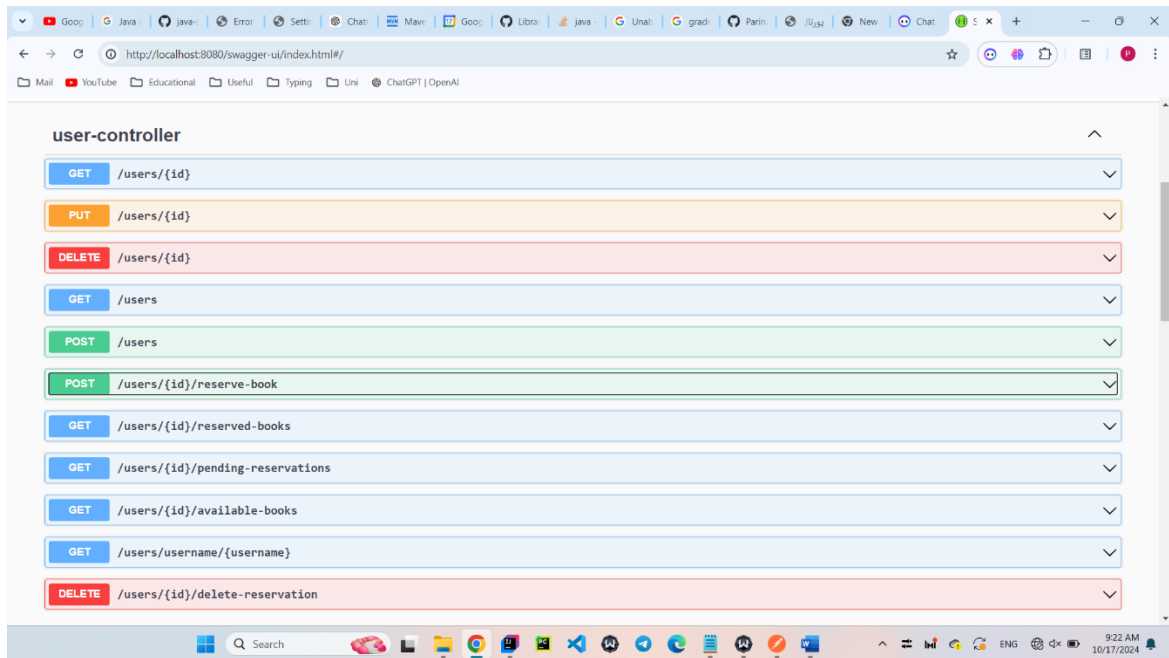
```
{
  "id": 1,
  "username": "john_doe",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@example.com",
  "hashedPassword": "SR+eVDhuQYcpN7cdnEwso42RQ581nXEP7XX2Gkdc8dgs=",
  "salt": "C0hxN171UCdv87CgX17/Q==",
  "role": "user",
  "name": "John Doe"
},
{
  "id": 2,
  "username": "jane_smith",
  "firstName": "Jane",
  "lastName": "Smith",
  "email": "jane.smith@example.com",
  "hashedPassword": "d2eL5P7t90Enpj1ZttM3K1Rt23fn@eIwsGpwOntpHBI=",
  "salt": "JwF+a6snHsjYzsStBvzi0Q==",
  "role": "user",
  "name": "Jane Smith"
}
```

اضافه کردن دیتابیس MongoDB برای جایگزینی استفاده از ArrayList

اضافه کردن قابلیت تعامل کاربر با ترمینال به صورتی که نتایج از API گرفته شده و برای او نشان داده شود.

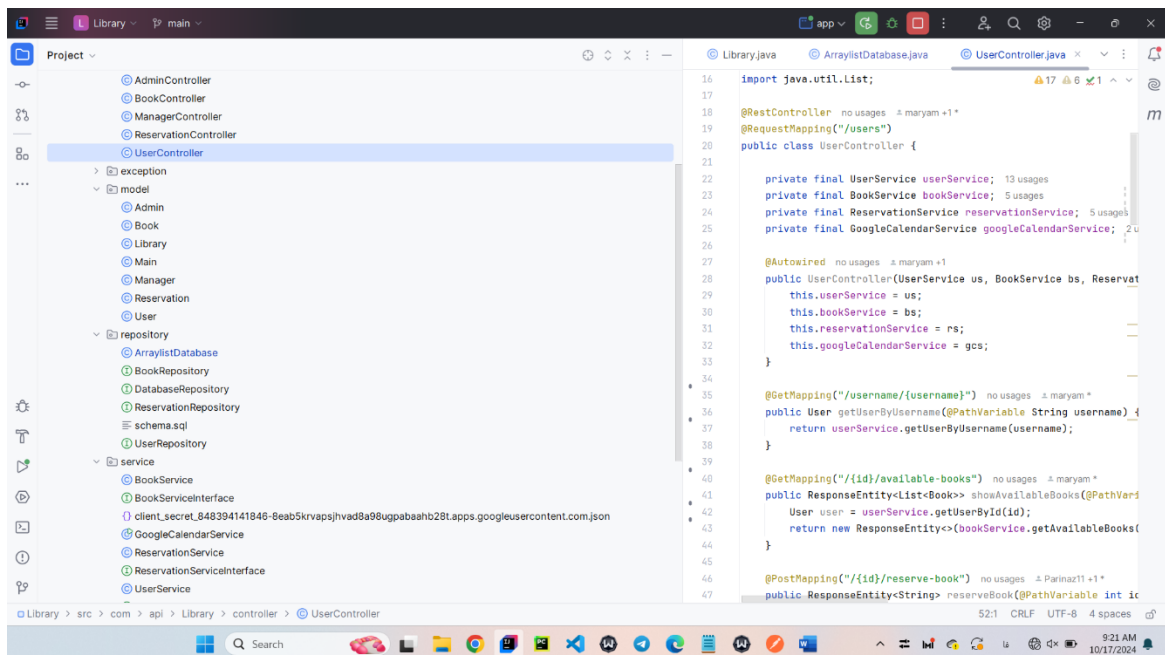
مرحله چهارم

اضافه کردن Swagger برای تست نتیجه پیاده سازی Rest API:



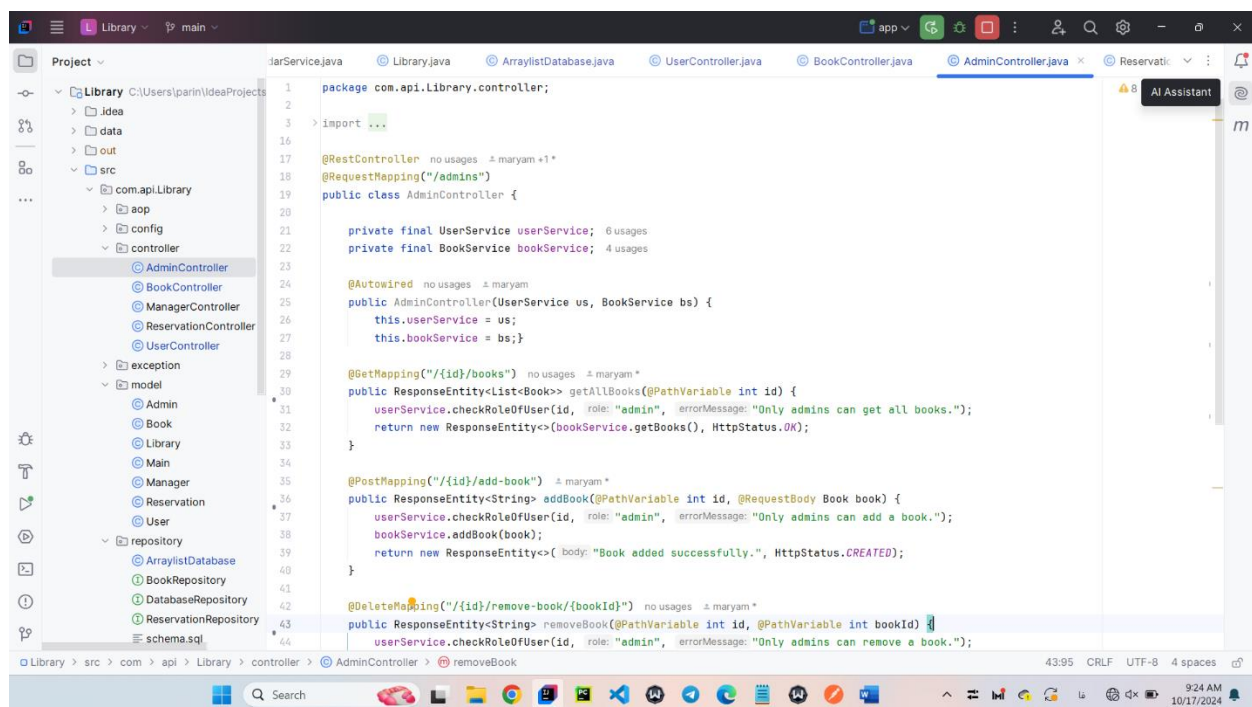
مرحله پنجم

تغییر ساختار پروژه به معماری 3-tier و پوشه بندی های Controller, Service, Mode, Repository.



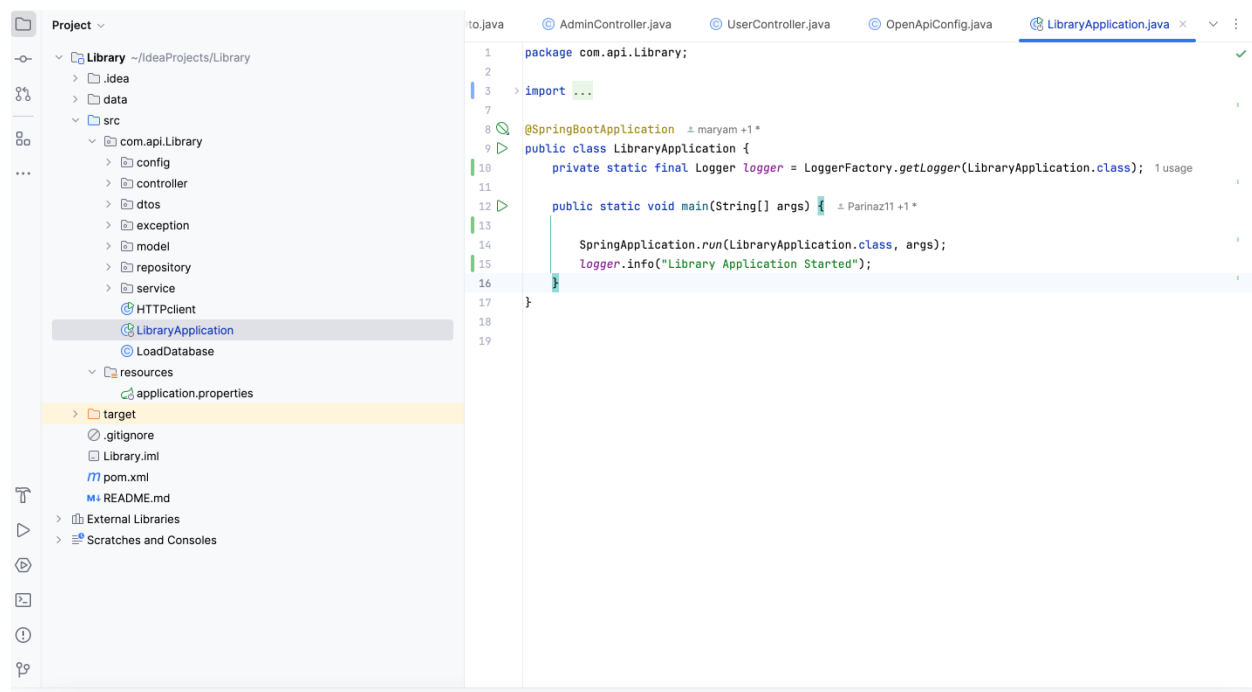
مرحله ششم

پیاده سازی Dependency Injection برای کلاس های مختلف پروژه:



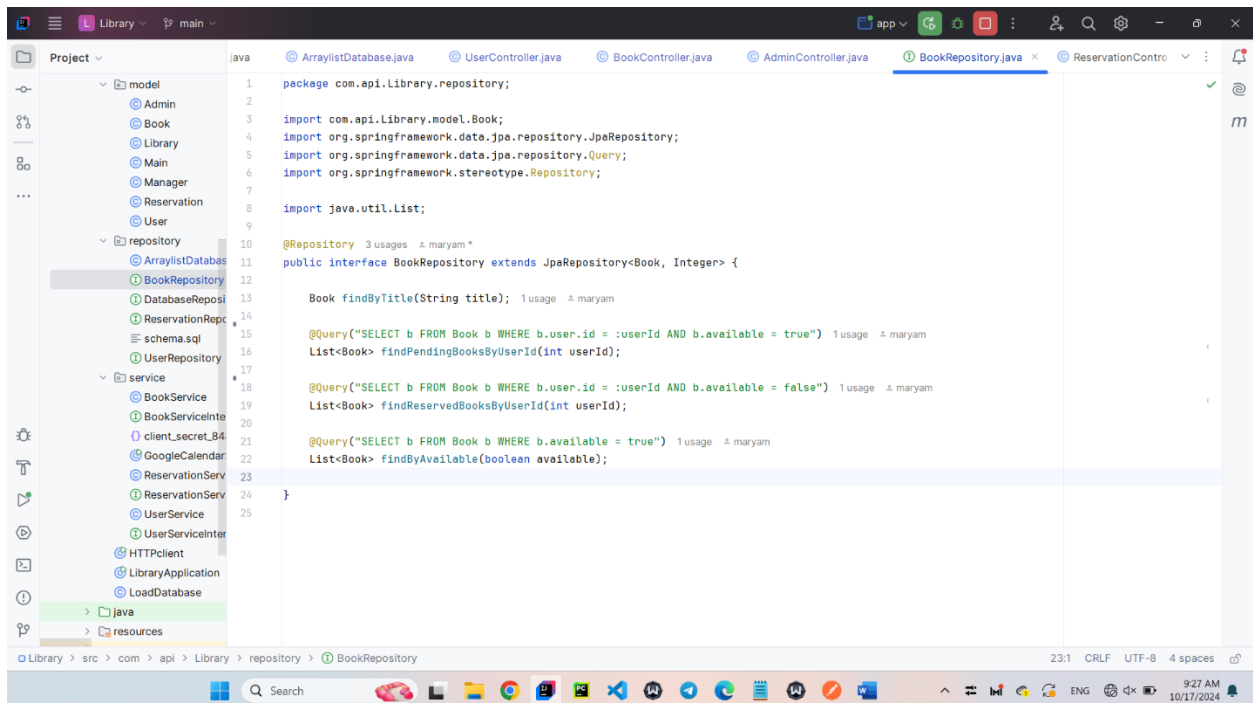
مرحله هفتم

اضافه کردن Log با j4sfl

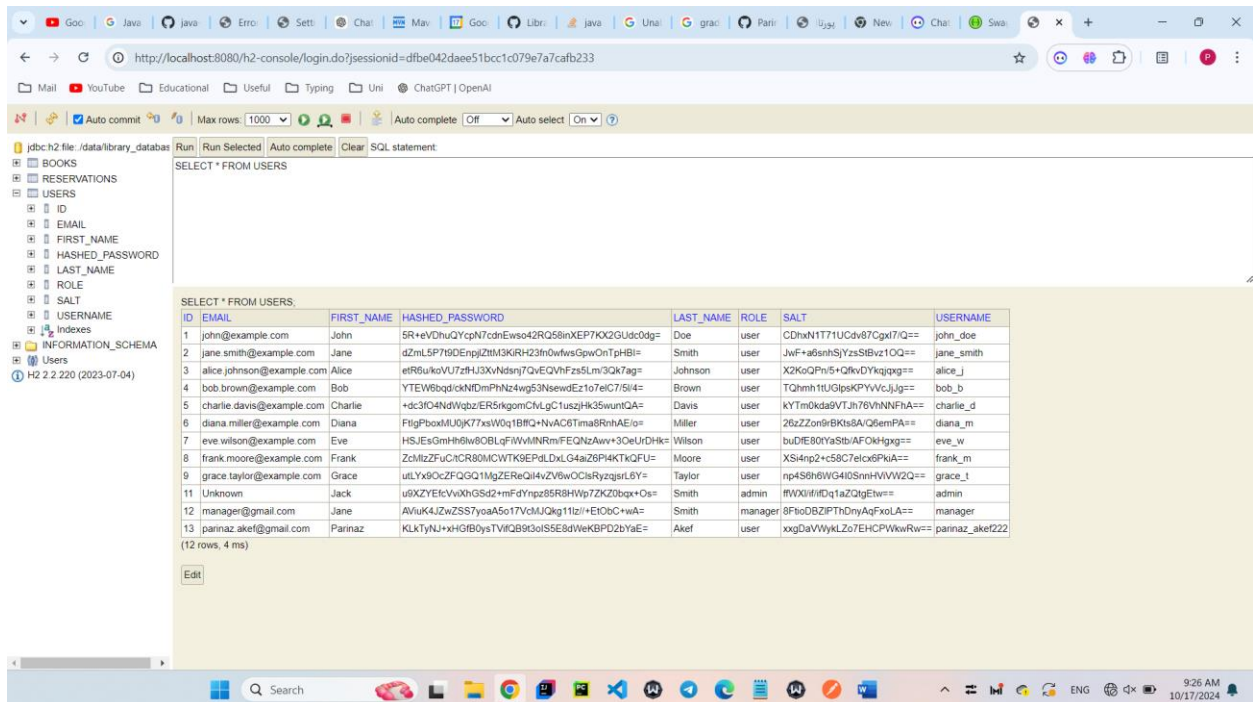


مرحله هشتم

پیاده سازی دیتابیس h2 و JPA Repository



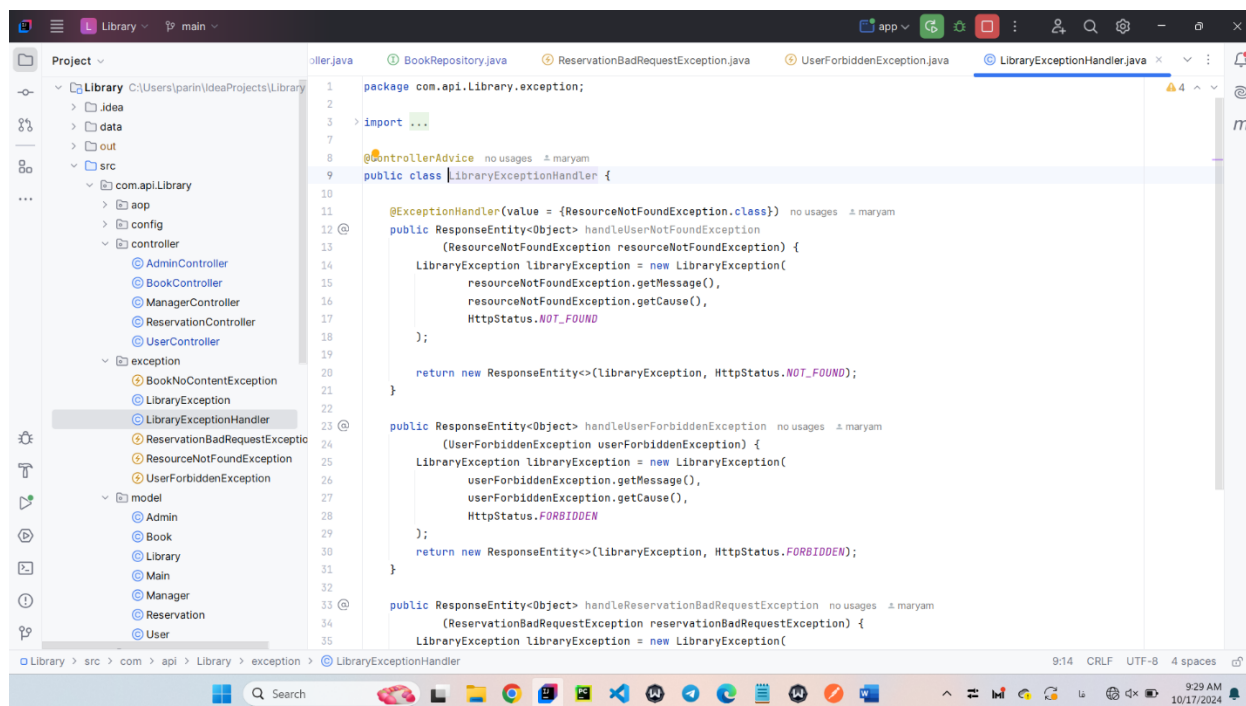
مشاهده پایگاه داده با استفاده از h2-console



مرحله نهم

اضافه کردن Custom Exceptions برای مدیریت خطاها، نمایش کد های HTTP مخصوص آنها و مدیریت Exception ها برای کاربر

کد Handler:



```
package com.api.library.exception;

import ...

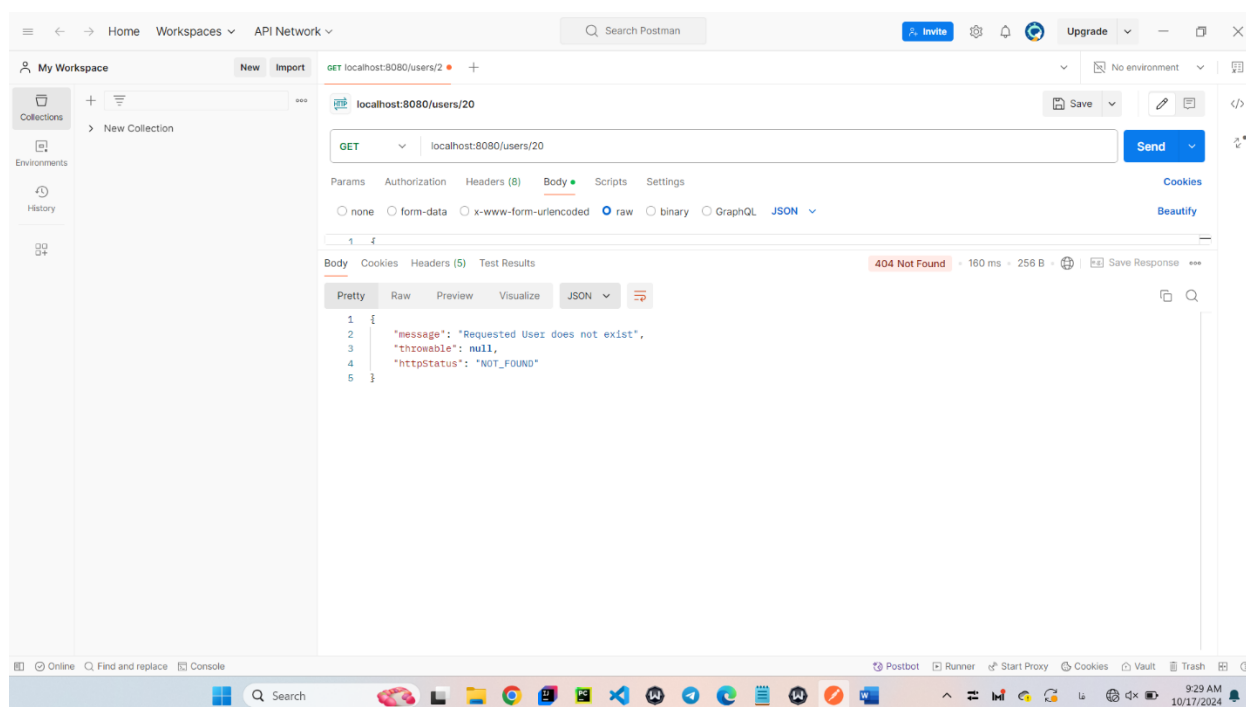
@ControllerAdvice
public class LibraryExceptionHandler {

    @ExceptionHandler(value = {ResourceNotFoundException.class})
    public ResponseEntity<Object> handleUserNotFoundException(
        ResourceNotFoundException resourceNotFoundException) {
        LibraryException libraryException = new LibraryException(
            resourceNotFoundException.getMessage(),
            resourceNotFoundException.getCause(),
            HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(value = {UserForbiddenException.class})
    public ResponseEntity<Object> handleUserForbiddenException(
        UserForbiddenException userForbiddenException) {
        LibraryException libraryException = new LibraryException(
            userForbiddenException.getMessage(),
            userForbiddenException.getCause(),
            HttpStatus.FORBIDDEN);
    }

    @ExceptionHandler(value = {ReservationBadRequestException.class})
    public ResponseEntity<Object> handleReservationBadRequestException(
        ReservationBadRequestException reservationBadRequestException) {
        LibraryException libraryException = new LibraryException(
            reservationBadRequestException.getMessage(),
            reservationBadRequestException.getCause(),
            HttpStatus.BAD_REQUEST);
    }
}
```

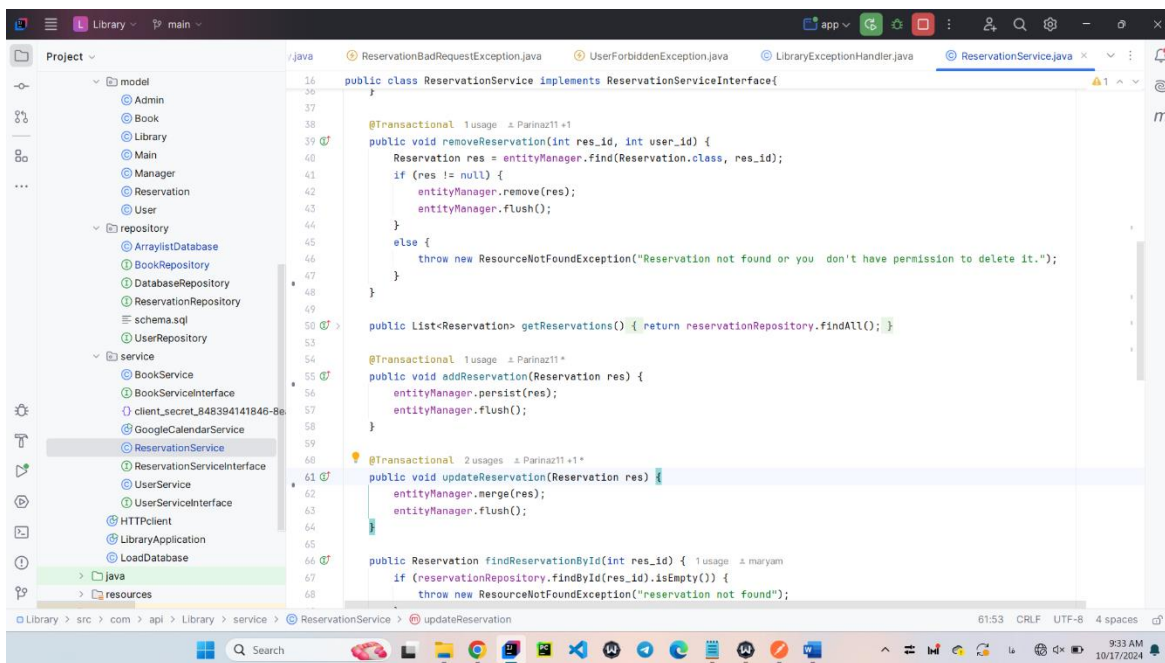
مشاهده نتیجه تست در Postman:



```
{
  "message": "Requested User does not exist",
  "throwable": null,
  "httpStatus": "NOT_FOUND"
}
```

مرحله دهم

پیاده سازی Entity Manager برای مدیریت session ها



مرحله یازدهم

اضافه کردن Security و Validation

```
@Component  
public class JwtAuthenticationFilter extends OncePerRequestFilter {  
    private final HandlerExceptionResolver handlerExceptionResolver;  
  
    private final JwtService jwtService;  
    private final UserDetailsService userDetailsService;  
  
    public JwtAuthenticationFilter(  
        JwtService jwtService,  
        UserDetailsService userDetailsService,  
        HandlerExceptionResolver handlerExceptionResolver  
    ) {  
        this.jwtService = jwtService;  
        this.userDetailsService = userDetailsService;  
        this.handlerExceptionResolver = handlerExceptionResolver;  
    }  
  
    @Override  
    protected void doFilterInternal(  
        @NonNull HttpServletRequest request,  
        @NonNull HttpServletResponse response,  
        @NonNull FilterChain filterChain  
    ) throws ServletException, IOException {  
        final String authHeader = request.getHeader("Authorization");  
  
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {  
            filterChain.doFilter(request, response);  
            return;  
        }  
  
        try {  
            final String jwt = authHeader.substring(7);  
            final String userEmail = jwtService.extractUsername(jwt);  
        }  
    }  
}
```



```

@Configuration  ⚡ zeynabT
public class ApplicationConfiguration {
    private final UserRepository userRepository;  2 usages 💡

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

    @Bean  ⚡ zeynabT
    UserDetailsService userDetailsService() {
        return username -> userRepository.findByEmail(username)
            .orElseThrow(() -> new UsernameNotFoundException("User not found"));
    }

    @Bean  ⚡ zeynabT
    BCryptPasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean  ⚡ zeynabT
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
        return config.getAuthenticationManager();
    }

    @Bean  ⚡ zeynabT
    AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(passwordEncoder());

        return authProvider;
    }
}

```

مرحله دوازدهم

جایگزینی Log های پیشین با Log های جدید که از AOP استفاده کرده اند:

```

@Aspect  @ maryam
@Configuration
public class LoggingAspect {

    private final Logger logger = LoggerFactory.getLogger(this.getClass()); 16 usages

    @After(value = "execution(* com.api.Library.service.UserService.*(..))" //Pointcut @ maryam
    public void after(JoinPoint joinPoint) //Advice
    {
        logger.info("After execution of {}", joinPoint);
    }

    @Before("execution(* com.api.Library.service.*(..))" //Pointcut @ maryam
    public void before(JoinPoint joinPoint) //Advice
    {
        logger.info("Before execution of {}", joinPoint);
    }

    @Around("execution(* com.api.Library.service.*(..))" @ maryam
    public Object log(ProceedingJoinPoint joinPoint) throws Throwable {
        logger.info("Before execution of {}", joinPoint);
        Object result = joinPoint.proceed();
        logger.info("After execution of {}", joinPoint);
        return result;
    }

    @AfterThrowing(pointcut = "execution(* com.api.Library.service.*(..))", throwing = "ex") @ maryam
    public void logAfterThrowing(Exception ex) {
        logger.error("An Exception has been throws: " + ex.getMessage(), ex);
    }
}

```

```

2024-10-17T09:48:29.880+03:30 INFO 20290 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
2024-10-17T09:48:29.932+03:30 INFO 20290 --- [nio-8080-exec-1] c.a.l.aop.LoggingAspect$$SpringCGLIB$$0 : Before execution of execution(User com.api.Library.service
.UserService.saveUser(User))
2024-10-17T09:48:29.933+03:30 INFO 20290 --- [nio-8080-exec-1] c.a.l.aop.LoggingAspect$$SpringCGLIB$$0 : Before execution of execution(User com.api.Library.service
.UserService.saveUser(User))
2024-10-17T09:48:29.966+03:30 INFO 20290 --- [nio-8080-exec-1] c.a.l.aop.LoggingAspect$$SpringCGLIB$$0 : Method executed successfully, result: com.api.Library.model
.User@fd2487a
2024-10-17T09:48:29.966+03:30 INFO 20290 --- [nio-8080-exec-1] c.a.l.aop.LoggingAspect$$SpringCGLIB$$0 : After execution of execution(User com.api.Library.service
.UserService.saveUser(User))
2024-10-17T09:48:29.966+03:30 INFO 20290 --- [nio-8080-exec-1] c.a.l.aop.LoggingAspect$$SpringCGLIB$$0 : After execution of execution(User com.api.Library.service
.UserService.saveUser(User))

```

مرحله سیزدهم

اضافه کردن تست های Custom با Unit Test

```

@DataJpaTest new *
public class UserRepositoryTest {

    @Autowired
    private UserRepository userRepository;
    User user; 4 usages

    @BeforeEach new *
    void setUp() {
        user = new User( username: "maryamM", firstName: "maryam", lastName: "mah", email: "maryam.mahdizadeh@gmail.com", password: "password123m");
        userRepository.save(user);
    }

    @AfterEach new *
    void tearDown() {
        user = null;
        userRepository.deleteAll();
    }

    @Test new *
    void testGetUserByUsername_Found() {
        User foundUser = userRepository.findByUsername("maryamM");
        assertEquals(foundUser.getUsername(), user.getUsername());
    }
}

```

مرحله آخر

اضافه کردن UI با استفاده از Google Calendar API:

