



Final Project Work

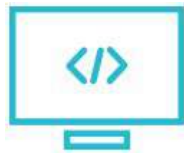
Points possible: 100

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete that were committed to for the given week.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

Continue contributing to your final project.



Screenshots of Code:

```
1 package com.promineotech.bookTracker;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @ComponentScan("com.promineotech.bookTracker")
7 @SpringBootApplication
8 public class App
9 {
10 {
11     public static void main( String[] args )
12     {
13         SpringApplication.run(App.class, args);
14     }
15 }
16 }
```

```
1 package com.promineotech.bookTracker.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 //import com.promineotech.bookTracker.service.BookService;
5
6 @RestController
7 @RequestMapping("users/{id}/authors")
8 public class AuthorController {
9
10     @Autowired
11     private AuthorService service;
12
13     // private BookService bookService;
14
15     @RequestMapping(value =("/{authorId}", method = RequestMethod.GET)
16     public ResponseEntity<Object> getAuthor(@PathVariable Long authorId) {
17         try {
18             return new ResponseEntity<Object>(service.getAuthorById(authorId), HttpStatus.OK);
19         } catch (Exception e) {
20             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
21         }
22     }
23
24     @RequestMapping(method = RequestMethod.GET)
25     public ResponseEntity<Object> getAuthors() {
26         return new ResponseEntity<Object>(service.getAuthors(), HttpStatus.OK);
27     }
28
29     // @RequestMapping(value =("/{authorId}/books")
30     // public ResponseEntity<Object> getBooks(){
31     //     return new ResponseEntity<Object>(bookService.getBooks(), HttpStatus.OK);
32     // }
33
34     @RequestMapping(method = RequestMethod.POST)
35     public ResponseEntity<Object> addAuthor(@RequestBody Author author) {
36         return new ResponseEntity<Object>(service.addAuthor(author), HttpStatus.CREATED);
37     }
38
39     @RequestMapping(value =("/{authorId}", method = RequestMethod.PUT)
40     public ResponseEntity<Object> updateAuthor(@RequestBody Author author, @PathVariable Long authorId) {
41         try {
42             return new ResponseEntity<Object>(service.updateAuthor(author, authorId), HttpStatus.OK);
43         } catch (Exception e) {
44             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
45         }
46     }
47
48     @RequestMapping(value =("/{authorId}", method = RequestMethod.DELETE)
49     public ResponseEntity<Object> removeAuthor(@PathVariable Long authorId) {
50         try {
51             service.removeAuthor(authorId);
52             return new ResponseEntity<Object>("Successfully removed author with id: " + authorId, HttpStatus.OK);
53         } catch (Exception e) {
54             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
55         }
56     }
57
58 }
```



PROMINEO TECH

```
1 package com.promineotech.bookTracker.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14
15 @RestController
16 @RequestMapping("users/{id}/books")
17 public class BookController {
18
19     @Autowired
20     private BookService service;
21
22     @RequestMapping(value="/{bookId}", method=RequestMethod.GET)
23     public ResponseEntity<Object> getBook(@RequestBody Long bookId) {
24         try {
25             return new ResponseEntity<Object>(service.getBookById(bookId), HttpStatus.OK);
26         } catch (Exception e) {
27             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
28         }
29     }
30
31     @RequestMapping(method=RequestMethod.GET)
32     public ResponseEntity<Object> getBooks() {
33         return new ResponseEntity<Object>(service.getBooks(), HttpStatus.OK);
34     }
35
36     @RequestMapping(method=RequestMethod.POST)
37     public ResponseEntity<Object> addBook(@RequestBody Book book, @PathVariable Long id) throws Exception {
38         return new ResponseEntity<Object>(service.addBook(book, id), HttpStatus.CREATED);
39     }
40
41     @RequestMapping(value="/{bookId}", method=RequestMethod.PUT)
42     public ResponseEntity<Object> updateBook(@RequestBody Book book, @PathVariable Long bookId) {
43         try {
44             return new ResponseEntity<Object>(service.updateBook(book, bookId), HttpStatus.OK);
45         } catch (Exception e) {
46             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
47         }
48     }
49
50     @RequestMapping(value="/{bookId}", method=RequestMethod.DELETE)
51     public ResponseEntity<Object> removeBook(@PathVariable Long id) {
52         try {
53             service.removeBook(id);
54             return new ResponseEntity<Object>("Successfully removed book with id: " + id, HttpStatus.OK);
55         } catch (Exception e) {
56             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
57         }
58     }
59 }
60
```




PROMINEO TECH

```
1 package com.promineotech.bookTracker.controller;
2
3 import java.nio.file.Files;
4
5 @RestController
6 @RequestMapping("/users")
7 public class UserController {
8
9     private static String UPLOADED_FOLDER = "./pictures/";
10
11     @Autowired
12     private UserService service;
13
14     @RequestMapping(value = "/register", method = RequestMethod.POST)
15     public ResponseEntity<Object> register(@RequestBody User user) {
16         return new ResponseEntity<Object>(service.createUser(user), HttpStatus.CREATED);
17     }
18
19     @RequestMapping(value = "/login", method = RequestMethod.POST)
20     public ResponseEntity<Object> login(@RequestBody User user) {
21         try {
22             return new ResponseEntity<Object>(service.login(user), HttpStatus.OK);
23         } catch (Exception e) {
24             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
25         }
26     }
27
28     @RequestMapping(method = RequestMethod.GET)
29     public ResponseEntity<Object> getUsers() {
30         return new ResponseEntity<Object>(service.getUsers(), HttpStatus.OK);
31     }
32
33     @RequestMapping(value =("/{id}", method = RequestMethod.GET)
34     public ResponseEntity<Object> getUser(@PathVariable Long id) {
35         try {
36             return new ResponseEntity<Object>(service.getUserById(id), HttpStatus.OK);
37         } catch (Exception e) {
38             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
39         }
40     }
41
42     @RequestMapping(value =("/{id}/profilePicture", method = RequestMethod.POST)
43     public ResponseEntity<Object> singleFileUpload(@PathVariable Long id, @RequestParam("file") MultipartFile file) {
44         if (file.isEmpty()) {
45             return new ResponseEntity<Object>("Please upload a file.", HttpStatus.BAD_GATEWAY);
46         }
47
48         try {
49             String url = UPLOADED_FOLDER + file.getOriginalFilename();
50             byte[] bytes = file.getBytes();
51             Path path = Paths.get(url);
52             Files.write(path, bytes);
53             return new ResponseEntity<Object>(service.updateProfilePicture(id, url), HttpStatus.CREATED);
54         } catch (Exception e) {
55             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
56         }
57     }
58 }
```



PROMINEO TECH

```
15 @Entity
16 public class Author {
17
18     private Long authorId;
19     private String firstName;
20     private String lastName;
21
22     @JsonIgnore
23     private Set<Book> books;
24
25     @JsonIgnore
26     private User user;
27
28     @Id
29     @GeneratedValue(strategy = GenerationType.AUTO)
30     public Long getAuthorId() {
31         return authorId;
32     }
33
34     public void setAuthorId(Long authorId) {
35         this.authorId = authorId;
36     }
37
38     public String getFirstName() {
39         return firstName;
40     }
41
42     public void setFirstName(String firstName) {
43         this.firstName = firstName;
44     }
45
46     public String getLastName() {
47         return lastName;
48     }
49
50     public void setLastName(String lastName) {
51         this.lastName = lastName;
52     }
53
54     @OneToMany(mappedBy = "author")
55     public Set<Book> getBooks() {
56         return books;
57     }
58
59     public void setBooks(Set<Book> books) {
60         this.books = books;
61     }
62
63     @ManyToOne
64     @JoinColumn(name = "userId")
65     public User getUser() {
66         return user;
67     }
68
69     public void setUser(User user) {
70         this.user = user;
71     }
72 }
```



PROMINEO TECH

```
1 package com.promineotech.bookTracker.entity;
2
3 import javax.persistence.Entity;
4
5 @Entity
6 public class Book {
7
8     private Long bookId;
9     private String title;
10    private String genre = "Thriller";
11    private Long publishYear;
12    private String readDate;
13
14    @Enumerated(EnumType.STRING)
15    private BookRating rating = BookRating.GD;
16
17    private Author author;
18
19    private User user;
20
21    @Id
22    @GeneratedValue(strategy = GenerationType.AUTO)
23    public Long getBookId() {
24        return bookId;
25    }
26
27    public void setBookId(Long bookId) {
28        this.bookId = bookId;
29    }
30
31    public String getTitle() {
32        return title;
33    }
34
35    public void setTitle(String title) {
36        this.title = title;
37    }
38
39    public String getGenre() {
40        return genre;
41    }
42
43    public void setGenre(String genre) {
44        this.genre = genre;
45    }
46
47    public Long getPublishYear() {
48        return publishYear;
49    }
50
51    public void setPublishYear(Long publishYear) {
52        this.publishYear = publishYear;
53    }
54
55 }
```



PROMINEO TECH

```
64 public String getReadDate() {
65     return readDate;
66 }
67
68 public void setReadDate(String readDate) {
69     this.readDate = readDate;
70 }
71
72 public BookRating getRating() {
73     return rating;
74 }
75
76 public void setRating(BookRating rating) {
77     this.rating = rating;
78 }
79
80 @ManyToOne
81 @JoinColumn(name = "authorId")
82 public Author getAuthor() {
83     return author;
84 }
85
86 public void setAuthor(Author author) {
87     this.author = author;
88 }
89
90 @ManyToOne
91 @JoinColumn(name = "userId")
92 public User getUser() {
93     return user;
94 }
95
96 public void setUser(User user) {
97     this.user = user;
98 }
99 }
100
```

```
1 package com.promineotech.bookTracker.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6 public interface AuthorRepository extends CrudRepository<Author, Long> {
7
8 }
9
10
```

```
1 package com.promineotech.bookTracker.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6 public interface BookRepository extends CrudRepository<Book, Long> {
7
8 }
9
10
```

```
1 package com.promineotech.bookTracker.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6 public interface UserRepository extends CrudRepository<User, Long> {
7
8     public User findByUsername(String username);
9 }
10
11
```




PROMINEO TECH

```
1 package com.promineotech.bookTracker.service;
2
3 import org.apache.logging.log4j.LogManager;
10
11 @Service
12 public class AuthorService {
13
14     private static final Logger logger = LogManager.getLogger(AuthorService.class);
15
16     @Autowired
17     private AuthorRepository repo;
18
19     public Author getAuthorById(Long id) throws Exception {
20         try {
21             return repo.findOne(id);
22         } catch (Exception e) {
23             logger.error("Exception occurred while trying to retrieve Author.");
24             throw e;
25         }
26     }
27
28     public Iterable<Author> getAuthors() {
29         return repo.findAll();
30     }
31
32     public Author addAuthor(Author author) {
33         return repo.save(author);
34     }
35
36     public Author updateAuthor(Author author, Long id) throws Exception {
37         try {
38             Author oldAuthor = repo.findOne(id);
39             oldAuthor.setFirstName(author.getFirstName());
40             oldAuthor.setLastName(author.getLastName());
41             oldAuthor.setBooks(author.getBooks());
42             return repo.save(oldAuthor);
43         } catch (Exception e) {
44             logger.error("Exception occurred while trying to update author.");
45             throw new Exception("Unable to update author.");
46         }
47     }
48
49     public void removeAuthor(Long id) throws Exception {
50         try {
51             repo.delete(id);
52         } catch (Exception e) {
53             logger.error("Exception occurred while trying to delete author.");
54             throw new Exception("Unable to remove author.");
55         }
56     }
57 }
58
```




PROMINEO TECH

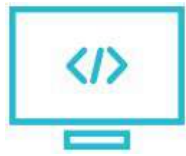
```
13 @Service
14 public class BookService {
15
16     private static final Logger logger = LogManager.getLogger(BookService.class);
17
18     @Autowired
19     private BookRepository repo;
20
21     public Iterable<Book> getBooks() {
22         return repo.findAll();
23     }
24
25     public Book getBookById(Long id) throws Exception {
26         try {
27             return repo.findOne(id);
28         } catch (Exception e) {
29             logger.error("Exception occurred while trying to retrieve book.");
30             throw e;
31         }
32     }
33
34     @Autowired
35     private AuthorRepository authorRepo;
36
37     public Book addBook(Book book, Long userId) throws Exception {
38         Author author = authorRepo.findOne(userId);
39         if (author == null) {
40             logger.error("Exception occurred while trying to add book.");
41             throw new Exception("Author does not exist in database.");
42         }
43         book.setAuthor(author);
44         return repo.save(book);
45     }
46
47     public Book updateBook(Book book, Long id) throws Exception {
48         try {
49             Book oldBook = repo.findOne(id);
50             oldBook.setAuthor(book.getAuthor());
51             oldBook.setGenre(book.getGenre());
52             oldBook.setPublishYear(book.getPublishYear());
53             oldBook.setRating(book.getRating());
54             oldBook.setReadDate(book.getReadDate());
55             oldBook.setTitle(book.getTitle());
56             return repo.save(oldBook);
57         } catch (Exception e) {
58             logger.error("Exception occurred while trying to update book.");
59             throw new Exception("Unable to update book.");
60         }
61     }
62
63     public void removeBook(Long id) throws Exception {
64         try {
65             repo.delete(id);
66         } catch (Exception e) {
67             logger.error("Exception occurred while trying to delete book: " + id, e);
68             throw new Exception("Unable to remove book.");
69         }
70     }
71 }
```



PROMINEO TECH

```
1 package com.promineotech.bookTracker.service;
2
3 import org.apache.logging.log4j.LogManager;
4
5
6
7
8
9
10
11 @Service
12 public class UserService {
13
14     private static final Logger logger = LogManager.getLogger(UserService.class);
15
16     @Autowired
17     private UserRepository repo;
18
19     public User createUser(User user) {
20         return repo.save(user);
21     }
22
23     public User login(User user) throws Exception {
24         User foundUser = repo.findByUsername(user.getUsername());
25         if (foundUser != null && foundUser.getPassword().equals(user.getPassword())) {
26             return foundUser;
27         } else {
28             throw new Exception("Invalid username or password.");
29         }
30     }
31
32     public User getUserById(Long id) throws Exception {
33         try {
34             return repo.findOne(id);
35         } catch (Exception e) {
36             logger.error("Exception occurred while trying to retrieve user: " + id, e);
37             throw e;
38         }
39     }
40
41     public Iterable<User> getUsers() {
42         return repo.findAll();
43     }
44
45     public User updateProfilePicture(Long userId, String url) throws Exception {
46         User user = repo.findOne(userId);
47         if (user == null) {
48             throw new Exception("User does not exist.");
49         }
50         user.setProfilePictureUrl(url);
51         return repo.save(user);
52     }
53 }
54
```

```
1 package com.promineotech.bookTracker.util;
2
3 public enum BookRating {
4
5     GD,
6     OK,
7     NG;
8
9 }
10
```



PROMINEO TECH

Screenshots of Running Application:

A screenshot of a REST client interface. The top bar shows a GET request to http://localhost:8080/users/1/books. Below the bar, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize BETA'. The 'Pretty' tab is selected, and the response is displayed in a JSON format. The JSON data represents a book and its associated author and user.

```
1 {
2   {
3     "bookId": 1,
4     "title": "Killing Floor",
5     "genre": "Thriller",
6     "publishYear": 1997,
7     "readDate": "10/09",
8     "rating": "GD",
9     "author": {
10      "authorId": 1,
11      "firstName": "Lee",
12      "lastName": "Child"
13    },
14    "user": {
15      "id": 1,
16      "username": "bryceha",
17      "profilePictureUrl": null
18    }
19  }
20 }
```

URL to GitHub Repository:

<https://github.com/bryceha/api-final>