

CS6.401 Software Engineering

Spring 2024

Project - 2

Team - 22

Gunank Singh Jakhar (2022201057)

M Elamparithy (2022801014)

Rishabh Gupta (2022202011)

Piyush Singh (2022201032)

Prashant Kumar (2022201058)

Feature 1: Better user management

Limitations: Only the admin can create a new account for a user. In addition, only the admin can edit and update user credentials.

Improvement: Enabled user self-registration.

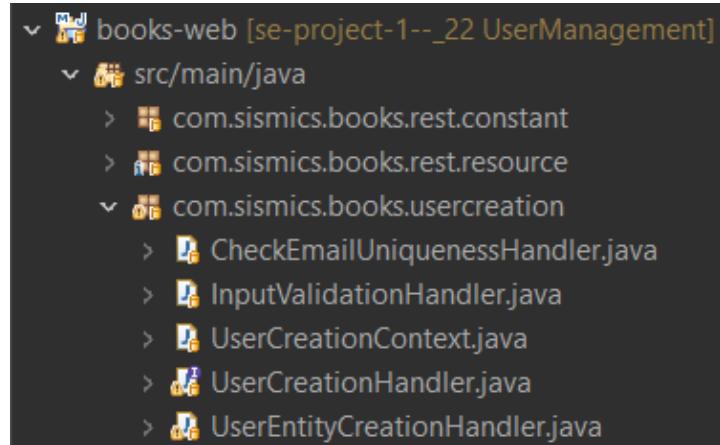
Design Pattern: Chain of Responsibility

The Chain of Responsibility is a behavioral design pattern that allows one to pass requests along a chain of handlers. Each handler processes the request or passes it on to the next handler in the chain. This design pattern decouples the sender of a request from its receivers, allowing multiple objects to handle the request. Each handler has a reference to the next handler in the chain.

The steps followed to register a user are:

1. Validation of Input (username and email uniqueness, password confirmation)
2. Creation of a User object with the inputs as the attributes
3. Saving the object in the database

Directory Structure



File Description

UserCreationContext.java (*Has all the input attributes, its object is passed as input to all the concrete handlers*)

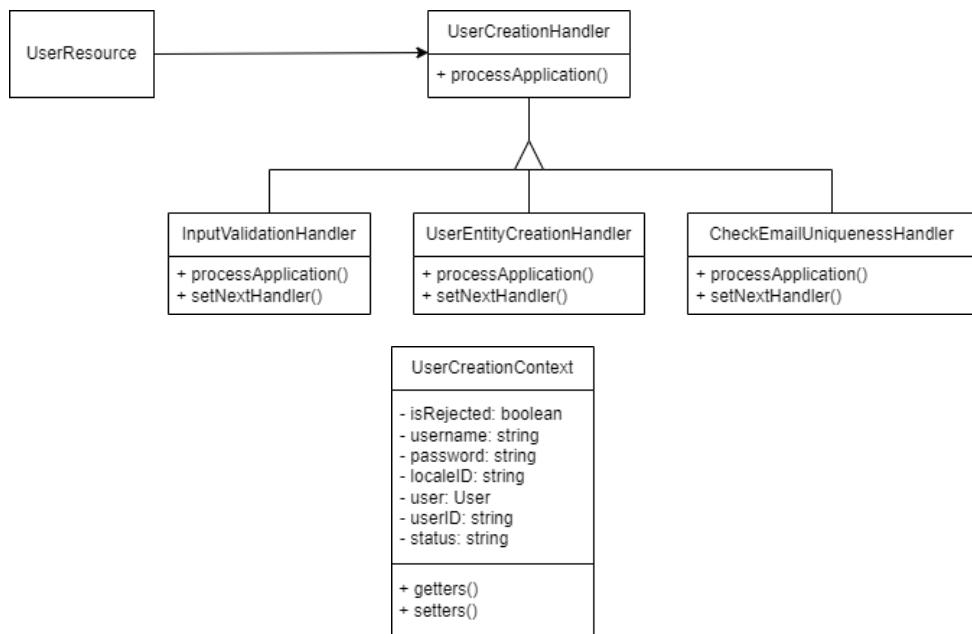
UserCreationHandler.java (*Handler Interface*)

InputValidationHandler.java (*validates username, email, and password*)

UserEntityCreationHandler.java (*creates user object and populates its attributes with the validated inputs*)

CheckEmailUniquenessHandler.java (*Checks if input email belongs to a registered user or not*)

UML Diagrams



Code Snippets

createAccount() function in **UserResource** class.

```
1 UserResource.java ×
2
3 57  @PUT
4 58  @Path("/create-account")
5 59  public Response createAccount(
6 60      @FormParam("username") String username,
7 61      @FormParam("password") String password,
8 62      @FormParam("locale") String localeId,
9 63      @FormParam("email") String email) throws Exception {
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65     UserCreationContext userCreationContext = new UserCreationContext(username,password,localeId,email);
66
67     //validate the input data
68     InputValidationHandler inputValidationHandler = new InputValidationHandler();
69     //create the user
70     UserEntityCreationHandler userEntityCreationHandler = new UserEntityCreationHandler();
71     //Raise a user creation event
72     CheckEmailUniquenessHandler checkEmailUniquenessHandler = new CheckEmailUniquenessHandler();
73
74     //setting up chain of responsibility
75     inputValidationHandler.setNextHandler(userEntityCreationHandler);
76     userEntityCreationHandler.setNextHandler(checkEmailUniquenessHandler);
77
78
79     inputValidationHandler.processApplication(userCreationContext);
80
```

UserCreationContext class

```
1 UserCreationContext.java ×
2
3
4
5
6 package com.sismics.books.usercreation;
7
8
9
10
11
12
13
14
15
16 public class UserCreationContext {
17     private boolean isRejected;
18     private String username;
19     private String password;
20     private String localeId;
21     private String email;
22     private User user;
23     private String userId;
24     private String status;
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80 }
```

UserCreationHandler class

```
1 UserCreationHandler.java ×
2
3
4
5
6 package com.sismics.books.usercreation;
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70 }
```

InputValidationHandler class

```
InputValidationHandler.java ×
1 package com.sismics.books.usercreation;
2
3 import com.sismics.rest.util.ValidationUtil;
4
5 public class InputValidationHandler implements UserCreationHandler{
6
7     private UserCreationHandler nextHandler;
8
9
10    public void setNextHandler(UserCreationHandler nextHandler) {
11        this.nextHandler = nextHandler;
12    }
13    @Override
14    public void processApplication(UserCreationContext userContext) throws Exception {
15
16        String username = userContext.getUsername();
17        String password = userContext.getPassword();
18        String localeId = userContext.getLocaleId();
19        String email = userContext.getEmail();
20
21        username = ValidationUtil.validateLength(username, "username", 3, 50);
22        ValidationUtil.validateAlphanumeric(username, "username");
23        password = ValidationUtil.validateLength(password, "password", 8, 50);
24        email = ValidationUtil.validateLength(email, "email", 3, 50);
25        ValidationUtil.validateEmail(email, "email");
26
27        userContext.setUsername(username);
28        userContext.setPassword(password);
29        userContext.setLocaleId(localeId);
30        userContext.setEmail(email);
31    }
32}
```

UserEntityCreationHandler class

```
1 UserEntityCreationHandler.java ×
2 package com.sismics.books.usercreation;
3 import java.util.Date;
4
5 public class UserEntityCreationHandler implements UserCreationHandler {
6     private UserCreationHandler nextHandler;
7
8     public void setNextHandler(UserCreationHandler nextHandler) {
9         this.nextHandler = nextHandler;
10    }
11
12    @Override
13    public void processApplication(UserCreationContext userContext) throws Exception {
14
15        String username = userContext.getUsername();
16        String password = userContext.getPassword();
17        String email = userContext.getEmail();
18
19        User user = userContext.getUser();
20
21        user.setRoleId(Constants.DEFAULT_USER_ROLE);
22        user.setUsername(username);
23        user.setPassword(password);
24        user.setEmail(email);
25        user.setCreateDate(new Date());
26        user.setLocaleId(Constants.DEFAULT_LOCALE_ID);
27
28        if(nextHandler != null) {
29            try {
30                nextHandler.processApplication(userContext);
31            } catch (Exception e) {
32                // TODO Auto-generated catch block
33                e.printStackTrace();
34            }
35        }
36    }
37}
```

CheckEmailUniquenessHandler class

```
1 package com.sismics.books.usercreation;
2
3 import com.sismics.books.core.dao.jpa.UserDao;
4
5
6 public class CheckEmailUniquenessHandler implements UserCreationHandler{
7     private UserCreationHandler nextHandler;
8
9
10    public void setNextHandler(UserCreationHandler nextHandler) {
11        this.nextHandler = nextHandler;
12    }
13
14
15    @Override
16    public void processApplication(UserCreationContext userContext) throws Exception {
17        User user = userContext.getUser();
18
19        // Create the user
20        UserDao userDao = new UserDao();
21
22        // check for email uniqueness
23        User u = userDao.checkIfEmailExists(user.getEmail());
24        if (u != null) {
25            userContext.setStatus("A User account already exists with the specified email");
26            throw new ServerException("A User account already exists with the specified email", null);
27        }
28
29        try {
30            userDao.create(user);
31        } catch (Exception e) {
32            if ("AlreadyExistingUsername".equals(e.getMessage())) {
33                userContext.setStatus("A User account already exists with the specfied username");
34                throw new ServerException("AlreadyExistingUsername", "Login already used", e);
35            } else {
36                throw new ServerException("UnknownError", "Unknown Server Error", e);
37            }
38        }
39    }
40}
```

Demo

Register

Username

Elamparthy

E-mail

m.elamparthy@research.iiit.ac.in

Password

.....

Too short

Password (confirm)

Password (confirm)

Password and password confirmation must match

Register

Register

Username

Elamparthy

E-mail

m.elamparthy@research.iiit.ac.in

Password

.....

Password (confirm)

.....

Register

localhost:8080 says

user created

OK

Login

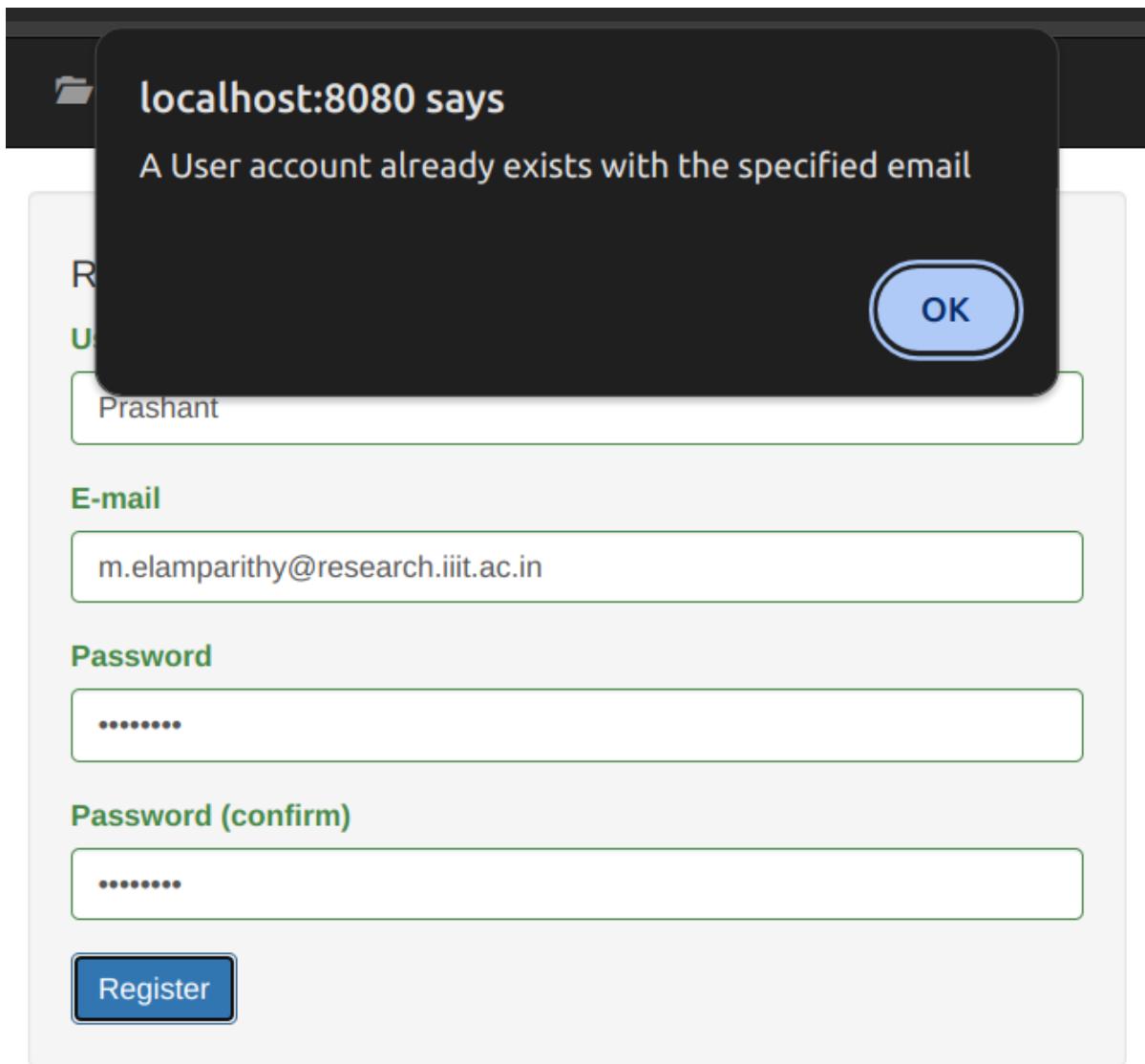
Username

Password

Remember me

Sign in

Not Registered? [Create an Account](#)



Feature 2: Common Library

To implement the common library feature, we have created a **CommonLibraryResource.java** file which has all the necessary functions to handle requests coming from the frontend.

CommonLibraryResource:

```
CommonLibraryResource.java ×
1 package com.sismics.books.rest.resource;
2
3+ import javax.ws.rs.Path;[]
43
44 @Path("commonlibrary")
45 public class CommonLibraryResource extends BaseResource {
46
47+     private void authenticateOrThrowForbidden() throws JSONException {[]
52
57+     public Response addBookInLibrary(){}
110
114+     public Response getAll() throws JSONException {[]
135
139+     public Response cover(){}
162
166+     public Response updateRating(){}
205
209+     public Response getById(){}
231
235+     public Response findBookByAuthor(){}
266
270+     public Response getAllAuthorName()throws JSONException {[]
289
293+     public Response findBookByGenre(){}
323
327+     public Response getAllGenreName()throws JSONException {[]
346
350+     public Response findBookByRating(){}
375
379+     public Response getData(@QueryParam("param") String param) throws JSONException {[]
418
419 }
420
```

Also, we made a separate table called “**T_COMMON_LIBRARY**” which signifies books added in the common library and class **CommonLibrary** for common library books to store all information and created **CommonLibraryDao** to perform **CRUD** operations.

CommonLibrary:

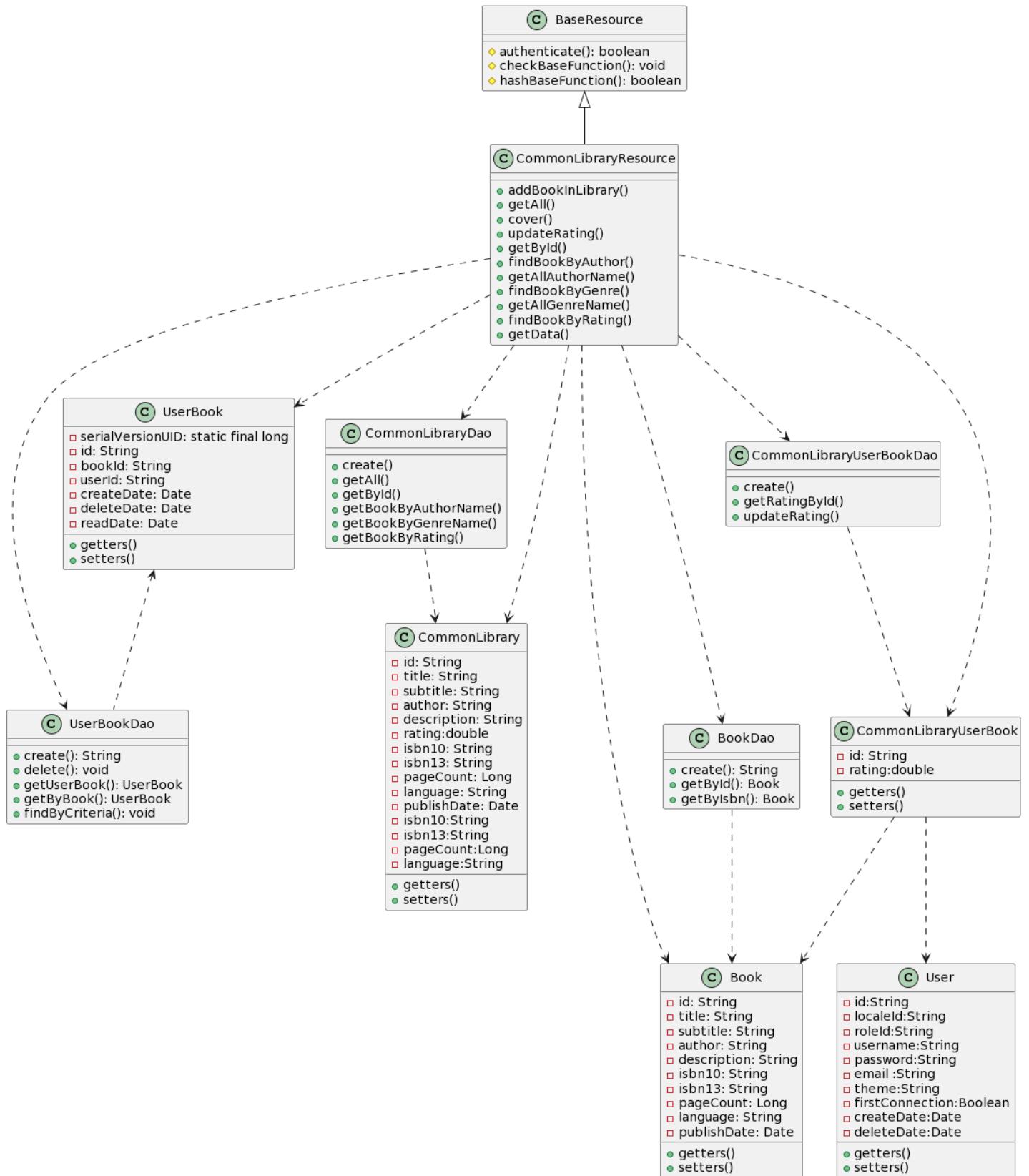
CommonLibrary.java ×

```
1 package com.sismics.books.core.model.jpa;
2
3+ import java.util.ArrayList;[]
11
12 @Entity
13 @Table(name = "T_COMMON_LIBRARY")
14 public class CommonLibrary {
15
18+     private String id;[]
19
21+     private String title;[]
22
24+     private String subtitle;[]
25
27+     private String author;[]
28
30+     private String description;[]
31
33+     private String genre;[]
34
36+     private double rating;[]
37
39+     private Integer noOfRatings;[]
40
42+     private String isbn10;[]
43
45+     private String isbn13;[]
46
48+     private Long pageCount;[]
49
51+     private String language;[]
52
54+     private Date publishDate;[]
55
```

CommonLibraryDao:

```
CommonLibraryDao.java ×
1 package com.sismics.books.core.dao.jpa;
2
3+ import java.util.List;[]
15
16 public class CommonLibraryDao {
17
18+     public String create(CommonLibrary commonLibrary) {[]
24
26+     public List<CommonLibrary> getAll(){[[
32
33+     public CommonLibrary getById(String id) {[[
41
43+     public List<CommonLibrary> getBookByAuthorName(String authorName) {[[
50
52+     public List<CommonLibrary> getBookByGenreName(String genreName) {[[
59
61+     public List<CommonLibrary> getBookByRating(double rating) {[[
68
69 }
70
```

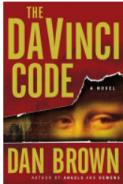
UML Diagram of Feature 2 Implementation:



Demo:

We added a button on each book **“Add To Library”** to add the particular book to the common library for public accessibility.

The Da Vinci Code Dan Brown



While in Paris on business, Harvard symbologist Robert Langdon receives an urgent late-night phone call: the elderly curator of the Louvre has been murdered inside the museum. Near the body, police have found a baffling cipher. While working to solve the enigmatic riddle, Langdon is stunned to discover it leads to a trail of clues hidden in the works of Da Vinci -- clues visible for all to see -- yet ingeniously disguised by the painter. Langdon joins forces with a gifted French cryptologist, Sophie Neveu, and learns the late curator was involved in the Priory of Sion -- an actual secret society whose members included Sir Isaac Newton, Botticelli, Victor Hugo, and Da Vinci, among others. In a breathless race through Paris, London, and beyond, Langdon and Neveu match wits with a faceless powerbroker who seems to anticipate their every move. Unless Langdon and Neveu can decipher the labyrinthine puzzle in time, the Priory's ancient secret -- and an explosive historical truth -- will be lost forever. THE DA VINCI CODE heralds the arrival of a new breed of lightning-paced, intelligent thriller...utterly unpredictable right up to its stunning conclusion.

Read book

Publication date 2003

ISBN 10 0385504209

ISBN 13 9780385504201

Number of pages 468

Language

Delete Edit Add To Library

When we click on the “**Add To Library**” Button, we take two values from user **Genres** and **Rating**, and then only add them to the common library area.

Genre(s):

Mystery, Detective fiction, Conspiracy fiction, Thriller

Rating:

8

Close Add Book

Here all authenticated users can see all books present in the common library.

Filter Books:

Authors Apply Genres Apply Rating Apply

 Average Rating Number of Ratings

Apply

Added Books

See all books



Sub-feature: 1 Book Details

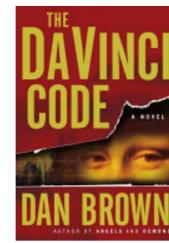
This is the view of books containing all the information like title, author, description, rating, ISBN, etc.

The Da Vinci Code

Dan Brown

Subtitle:

Description: While in Paris on business, Harvard symbologist Robert Langdon receives an urgent late-night phone call: the elderly curator of the Louvre has been murdered inside the museum. Near the body, police have found a baffling cipher. While working to solve the enigmatic riddle, Langdon is stunned to discover it leads to a trail of clues hidden in the works of Da Vinci -- clues visible for all to see -- yet ingeniously disguised by the painter. Langdon joins forces with a gifted French cryptologist, Sophie Neveu, and learns the late curator was involved in the Priory of Sion -- an actual secret society whose members included Sir Isaac Newton, Botticelli, Victor Hugo, and Da Vinci, among others. In a breathless race through Paris, London, and beyond, Langdon and Neveu match wits with a faceless powerbroker who seems to anticipate their every move. Unless Langdon and Neveu can decipher the labyrinthine puzzle in time, the Priory's ancient secret -- and an explosive historical truth -- will be lost forever. THE DA VINCI CODE heralds the arrival of a new breed of lightning-paced, intelligent thriller...utterly unpredictable right up to its stunning conclusion.

Published Date: 2003-03-18 00:00:00.0**Genre:** Mystery, Detective fiction, Conspiracy fiction, Thriller**Rating:** 8**Number of Ratings:** 1**ISBN-10:** 0385504209**ISBN-13:** 9780385504201**Page Count:** 468**Language:** 

Back

Sub-feature: 2 User Management

To resolve the issue of giving ratings multiple times to the same books in the common library, We created a new table “**T_COMMON_LIBRARY_USERBOOK**” which is used to store userId and bookId as primary key and rating as another attribute. The use of this table to check whether the user has given a rating or not.

CommonlibraryUserBook:

```
J CommonlibraryUserBook.java ×
1 package com.sismics.books.core.model.jpa;
2
3+ import javax.persistence.Column;[]
7
8 @Entity
9 @Table(name = "T_COMMON_LIBRARY_USERBOOK")
10 public class CommonlibraryUserBook {
11@  @Id
12  @Column(name = "CL_USERBOOK_ID", length = 255)
13  private String id;
14
15@  @Column(name = "CL_USERBOOK_RATING")
16  private double rating;
17}
```

CommonLibraryDao to perform **CRUD** operations on the database.

CommonlibraryUserBookDao.java

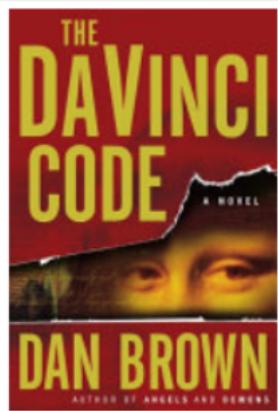
```
1 package com.sismics.books.core.dao.jpa;
2
3+ import javax.persistence.EntityManager;[]
10
11 public class CommonlibraryUserBookDao {
12
13
14+    public String create(CommonlibraryUserBook commonlibraryUserBook) {[]
15
16
17+    public CommonlibraryUserBook getRatingById(String userBookKey) {[]
18
19
20+    public void updateRating(CommonlibraryUserBook commonlibraryUserBook) {[]
21
22
23
24
25
26
27
28
29
30+    public void updateRating(CommonlibraryUserBook commonlibraryUserBook) {[]
31
32
33
34
35
36
37
38
39
40 }
41 }
```

Any user can rate any book present in the common library.

But If the same user gives a rating to the same book, then it will be modified.

otherwise, the rating will be changed accordingly. (we are showing the average rating here)

Added Books



Title: The Da Vinci Code

Author: Dan Brown

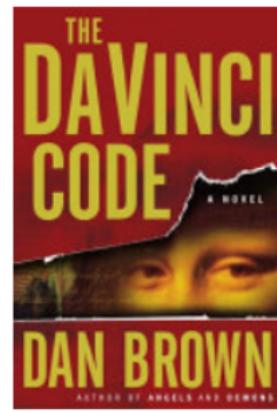
Genre: Mystery, Detective fiction, Co

Rating: 8

9

Rate

Added Books



Title: The Da Vinci Code

Author: Dan Brown

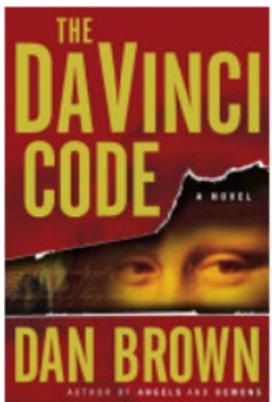
Genre: Mystery, Detective fiction, Co

Rating: 9

0

Rate

Added Books



Title: The Da Vinci Code

Author: Dan Brown

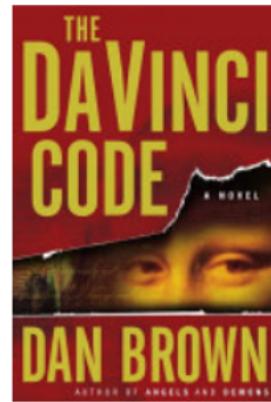
Genre: Mystery, Detective fiction, Co

Rating: 9

5

Rate

Added Books



Title: The Da Vinci Code

Author: Dan Brown

Genre: Mystery, Detective fiction, Co

Rating: 7

0

Rate

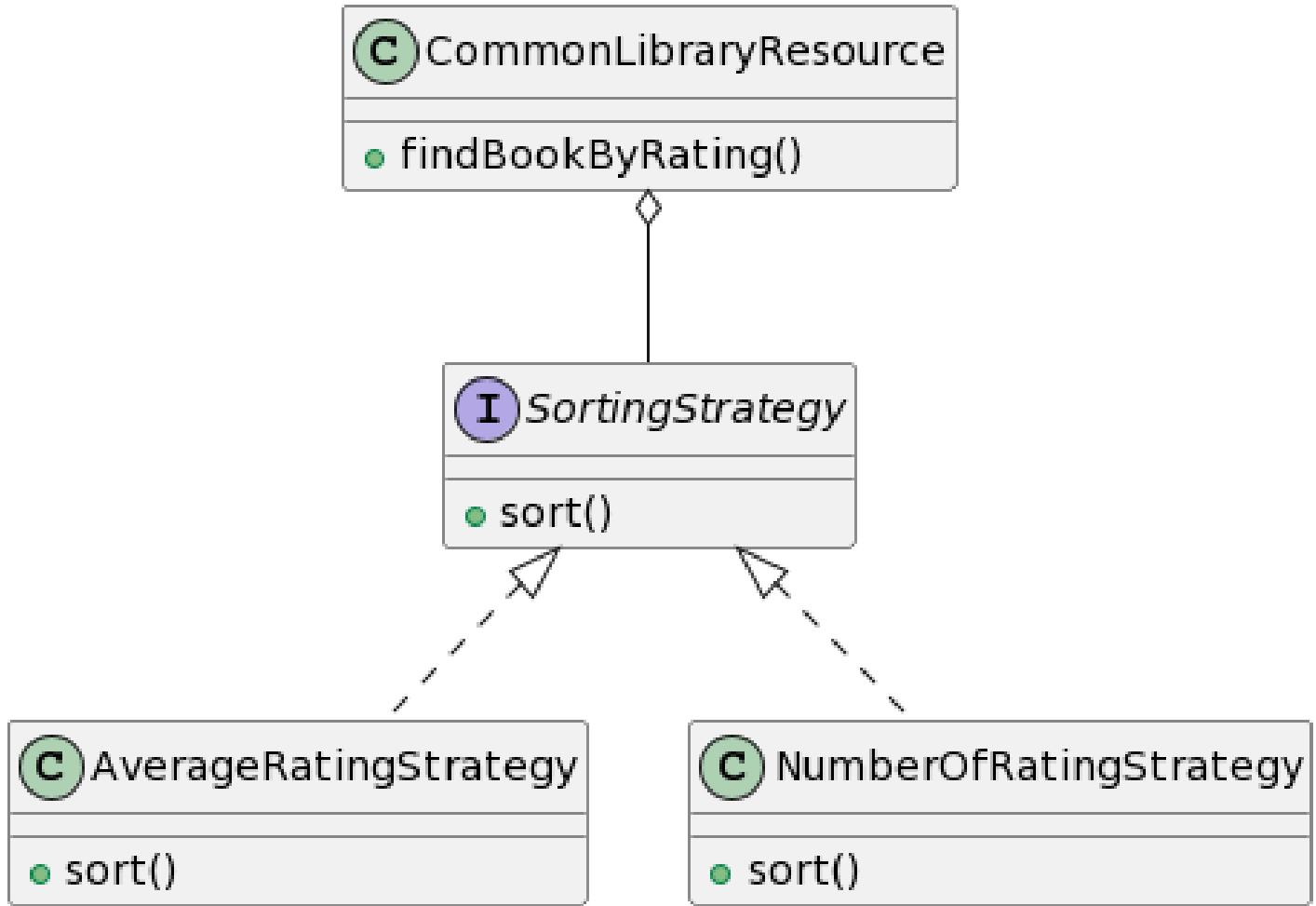
Sub-feature: 3 Book Ranking

Design Pattern: Strategy Design Pattern

The Strategy pattern is a behavioral design pattern that enables you to define a family of algorithms, encapsulate each one as an object, and make them interchangeable. This pattern allows the algorithms to vary independently from the clients that use them, making it easy to add new algorithms or modify existing ones without changing the clients.

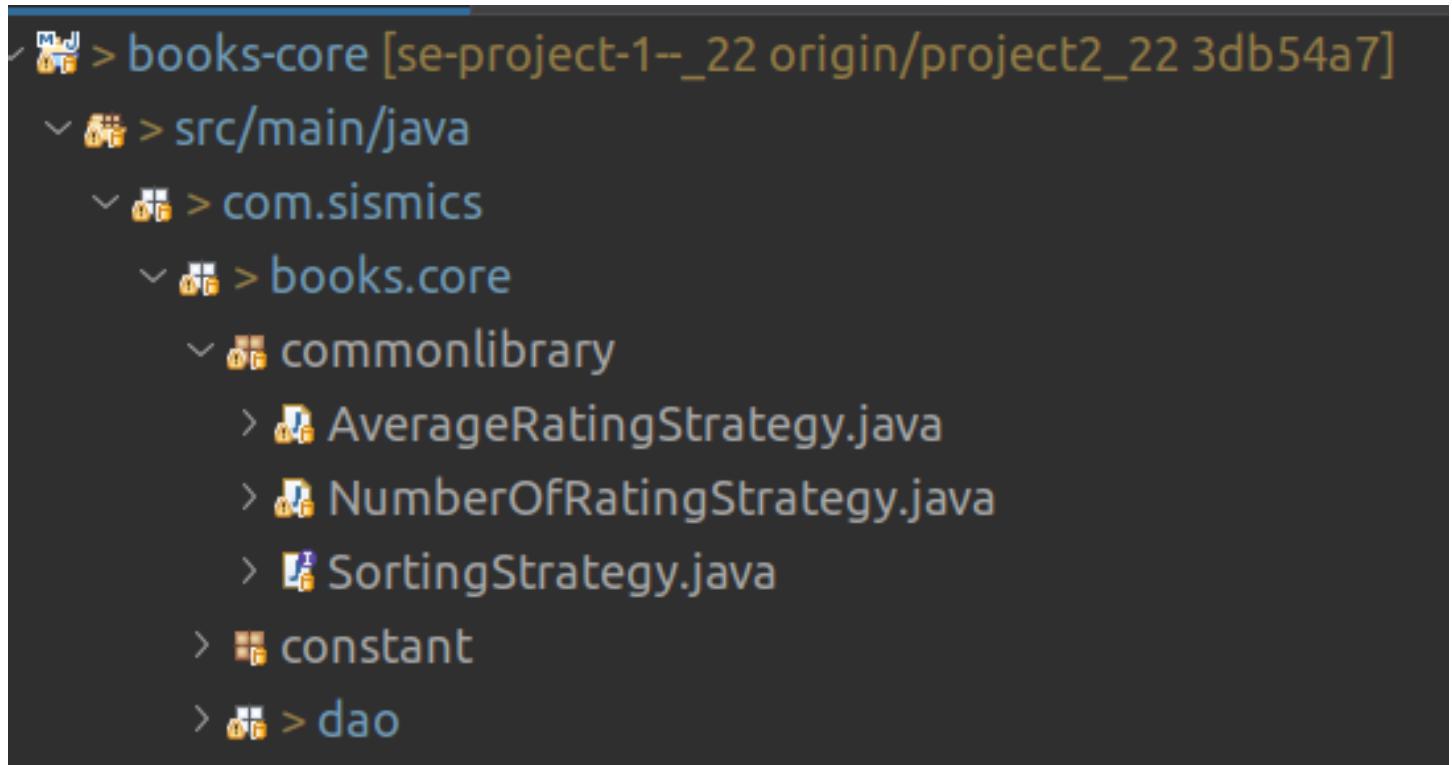
In this pattern, the strategy pattern defines a common interface for all the algorithms, allowing them to be used interchangeably. Each algorithm is encapsulated as a separate object that implements the common interface.

UML Diagram of Implementation:



CommonLibraryResource class serves the role of Context class in the Strategy pattern and uses the object of **SortingStrategy**. In this class, depending on the ranking condition (Sort by number of ratings or by average rating), the objects of **AverageRatingStrategy** and **NumberOfRatingStrategy** are assigned to the **SortingStrategy** object.

Directory Structure:



SortingStrategy - This abstract declares the sort method for the implementation.

AverageRatingStrategy - This class implements the sort method which sorts according to the average rating of each book.

NumberOfRatingStrategy - This class implements the sort method which sorts according to the number of ratings corresponding to a particular book.

SortingStrategy:

```
SortingStrategy.java ×
1 package com.sismics.books.core.commonlibrary;
2
3+import java.util.List;
4
5
6
7 public interface SortingStrategy {
8     public List<CommonLibrary> sort(List<CommonLibrary> listAllBooks);
9 }
10
```

The image shows a code editor with a dark theme. The title bar says 'SortingStrategy.java ×'. The code itself is a Java interface definition. It starts with a package declaration 'com.sismics.books.core.commonlibrary'. Then there's a blank line. Following that is an import statement 'import java.util.List;'. Another blank line follows. The interface 'SortingStrategy' is defined with a single method 'sort' that takes a list of 'CommonLibrary' objects and returns a list of 'CommonLibrary' objects. The code ends with a closing brace for the interface and a final blank line.

NumberOfRatingStrategy:

```
1 package com.sismics.books.core.commonlibrary;
2
3+ import java.util.ArrayList;□
9
10 public class NumberOfRatingStrategy implements SortingStrategy {
11
12     public List<CommonLibrary> sort(List<CommonLibrary> books) {
13
14         Collections.sort(books, new Comparator<CommonLibrary>() {
15             @Override
16             public int compare(CommonLibrary book1, CommonLibrary book2) {
17                 return -Double.compare(book1.getNoOfRatings(), book2.getNoOfRatings());
18             }
19         });
20         return books;
21     }
22 }
23
```

AverageRatingStrategy:

```
1
2 package com.sismics.books.core.commonlibrary;
3
4+ import java.util.ArrayList;□
11
12 public class AverageRatingStrategy implements SortingStrategy {
13     public List<CommonLibrary> sort(List<CommonLibrary> books) {
14
15         Collections.sort(books, new Comparator<CommonLibrary>() {
16             @Override
17             public int compare(CommonLibrary book1, CommonLibrary book2) {
18                 return -Double.compare(book1.getRating(), book2.getRating());
19             }
20         });
21         return books;
22     }
23 }
24
```

For the book ranking part, users can see the top 10 books based on average ratings or number of ratings.

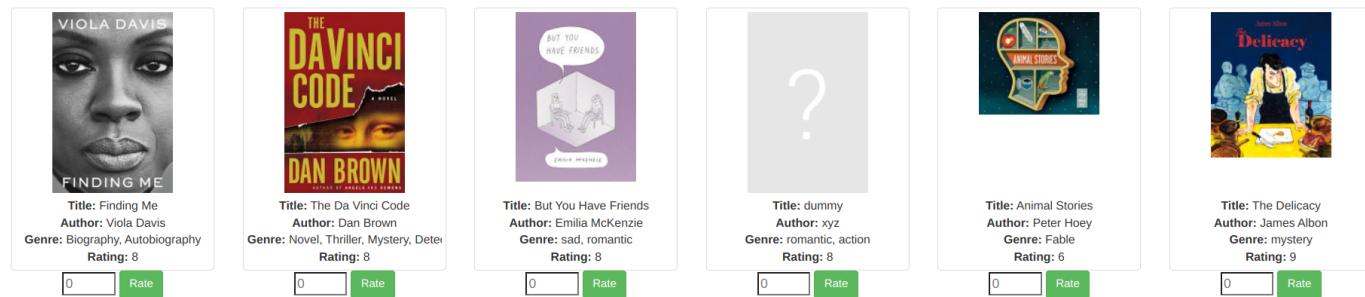
Filter Books:

[Authors](#) [Apply](#) [Genres](#) [Apply](#) [Rating](#) [Apply](#)

Average Rating
 Number of Ratings
[Apply](#)

Added Books

[See all books](#)



Showing top books based on average ratings.

Filter Books:

[Authors](#) [Apply](#) [Genres](#) [Apply](#) [Rating](#) [Apply](#)

Average Rating
 Number of Ratings
[Apply](#)

Added Books

[See all books](#)

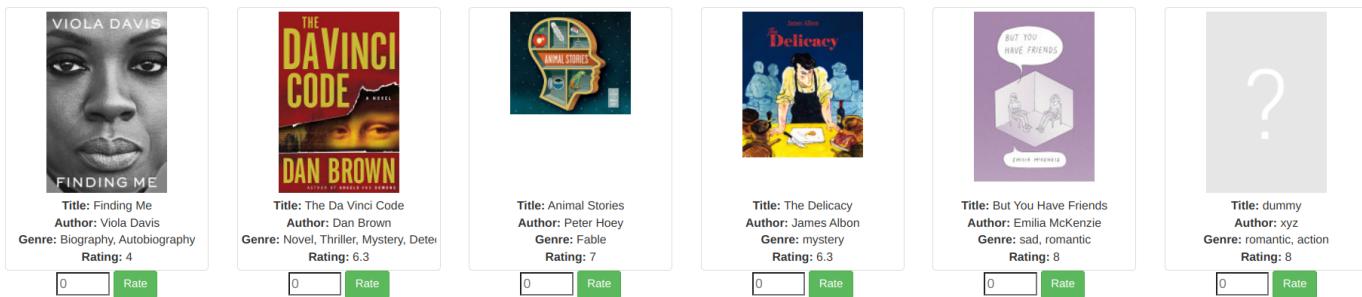


Showing top books based on the number of rating

Filter Books:

 Average Rating Number of Ratings

Added Books



Sub-feature: 4 Filtering

Users can also filter out books based on multiple authors, multiple genres, and a single rating.

Based on the selected options we show only those books in the UI.

Filter Books:

All Genres:

Biography
Autobiography
Novel
Thriller

- Autobiography
- Thriller
- Novel

Added Books

FINDING ME
Title: Finding Me
Author: Viola Davis
Genre: Biography, Autobiography
Rating: 8

THE DAVINCI CODE
Title: The Da Vinci Code
Author: Dan Brown
Genre: Novel, Thriller, Mystery, Detective
Rating: 8

Filter Books:

All Authors:

Emilia McKenzie
xyz
Peter Hoey
James Albon

- Peter Hoey
- James Albon

Added Books

ANIMAL STORIES
Title: Animal Stories
Author: Peter Hoey
Genre: Fable
Rating: 6

THE DELICACY
Title: The Delicacy
Author: James Albon
Genre: mystery
Rating: 9

Filter Books:

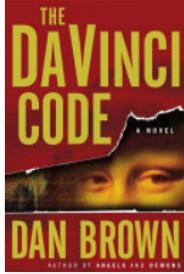
Authors Genres Rating

Ratings:

>6
=>7
>8
>9

- >7

Added Books



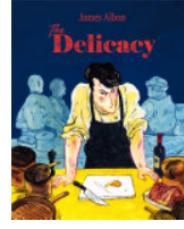
Title: The Da Vinci Code
Author: Dan Brown
Genre: Novel, Thriller, Mystery, Detective
Rating: 9



Title: But You Have Friends
Author: Emilia McKenzie
Genre: sad, romantic
Rating: 8



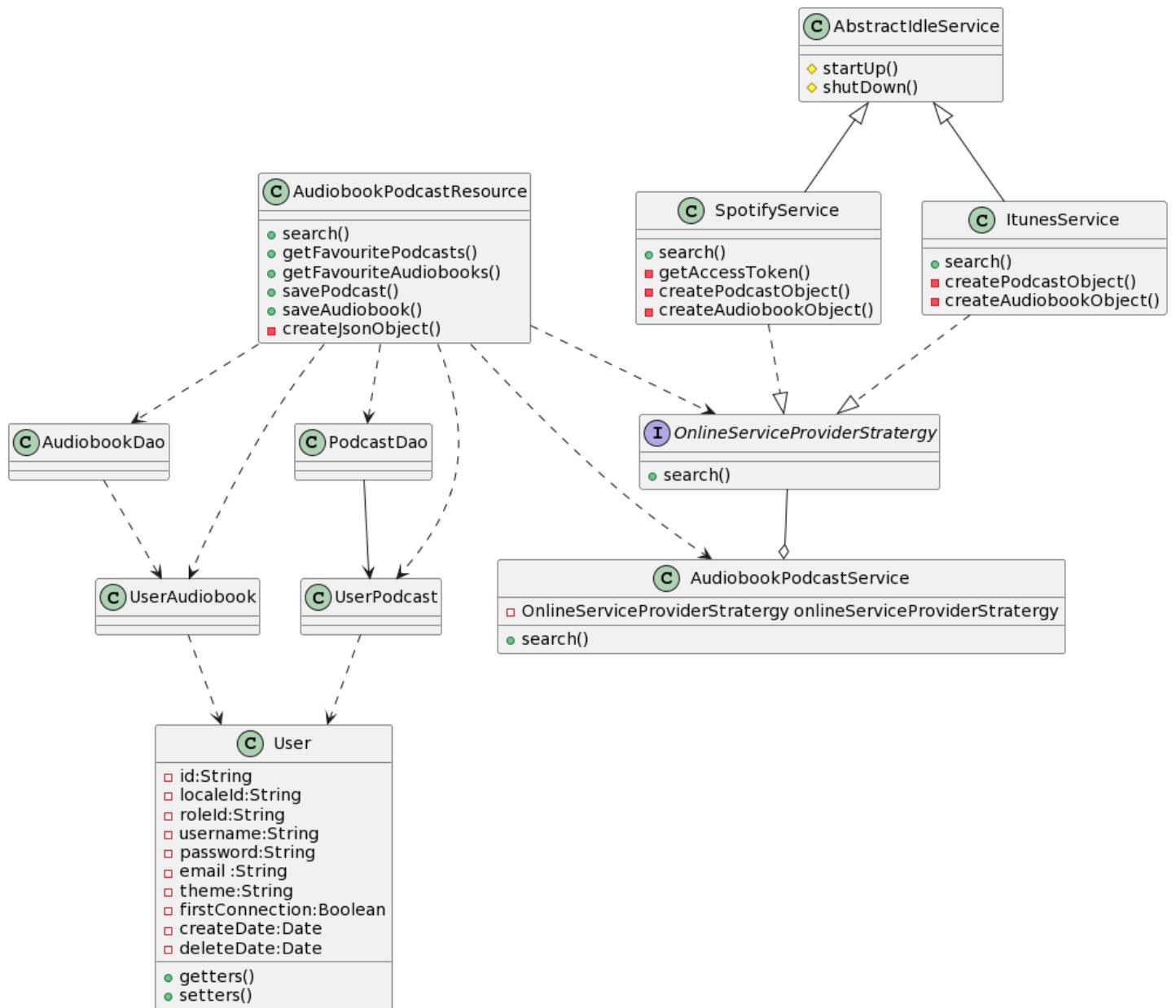
Title: dummy
Author: xyz
Genre: romantic, action
Rating: 8



Title: The Delicacy
Author: James Albon
Genre: mystery
Rating: 7.5

Feature 3: Online Integration

UML Diagram of Feature 3:



UserAudiobook: This class corresponds to the entity for the table (“T_AUDIOBOOK”) that stores all the audiobooks (whether obtained from Spotify or iTunes) favourited by a user.

UserAudiobook.java ×

```
15 @Entity
16 @Table(name = "T_AUDIOBOOK")
17 public class UserAudiobook {
19+     * UserAudiobook ID.□
23+     private String id;□
24
26+     * User ID.□
29+     private String userId;□
30
32+     * UserAudiobook Author.□
35+     private String author;□
36
38+     * UserAudiobook Preview Url.□
41+     private String previewUrl;□
42
44+     * UserAudiobook Image Url.□
47+     private String imageUrl;□
48
50+     * UserAudiobook Name□
53+     private String name;□
54
56+     * UserAudiobook Description.□
59+     private String description;□
60
62+     * UserAudiobook Narrator.□
65+     private String narrator;□
66
68+     * UserPodcast Provider□
71+     private String provider;□
72
```

AudiobookDao: This class is used to perform database queries on “T_AUDIOBOOK” table.

```
 AudiobookDao.java X
1 package com.sismics.books.core.dao.jpa;
2
3+import java.util.List;[]
11
12 /**
13  * Tag DAO.
14 *
15  * @author bqamard
16 */
17 public class AudiobookDao {
19+     * Creates a new audiobook. []
25+     public String create(UserAudiobook audiobook) { []
32+         * Gets a audiobook by its ID. []
37+     public UserAudiobook getById(String id) { []
45
47+         * Returns the list of all audiobook. []
52+     public List<UserAudiobook> getAllAudiobooks(String userId, String provider) { []
59
60 }
61
```

UserPodcast: This class corresponds to the entity for the table (“T_PODCAST”) that stores all the podcasts (whether obtained from Spotify or iTunes) favoured by a user.

```
1 package com.sismics.books.core.model.jpa;
2
3+import javax.persistence.Column;□
9
10 @Entity
11 @Table(name = "T_PODCAST")
12 public class UserPodcast{
14+    * UserPodcast ID.□
18+    private String id;□
19
21+    * User ID.□
24+    private String userId;□
25
27+    * UserPodcast Audio Url.□
30+    private String audioUrl;□
31
33+    * UserPodcast Image Url.□
36+    private String imageUrl;□
37
39+    * UserPodcast Name□
42+    private String name;□
43
45+    * UserPodcast Provider□
48+    private String provider;□
49
50
```

PodcastDao: This class is used to perform database queries on “T_PODCAST” table.

```
PodcastDao.java ×
1 package com.sismics.books.core.dao.jpa;
2
3+ import java.util.List;[]
11
12 /**
13  * Tag DAO.
14 *
15  * @author bgamard
16 */
17 public class PodcastDao {
19+     * Creates a new podcast.[]
25+     public String create(UserPodcast podcast) {[]
32+         * Gets a podcast by its ID.[]
37+     public UserPodcast getById(String id) {[]
45
47+         * Returns the list of all podcasts.[]
52+     public List<UserPodcast> getAllPodcasts(String userId, String provider) {[]
59
60 }
61
```

AbstractIdleService: com.google.common.util.concurrent.AbstractIdleService class. Basically, throughout the codebase this class is used wherever we want a singleton class (e.g. some services). Thus, it made sense for concrete strategy classes to inherit from this class.

OnlineServiceProviderStrategy: corresponding to the Strategy interface class in the Strategy pattern.

```
OnlineServiceProviderStrategy.java ×
1 package books-core/src/main/java/com/sismics/books/core/service/OnlineServiceProviderStrategy.java
2
3+ import java.io.IOException;[]
6
7 public interface OnlineServiceProviderStrategy{
8     public ArrayNode search(String query, String type) throws IOException;
9     public void saveToFavourites();
10 }
```

SpotifyService: corresponding to the concrete Strategy classes in the Strategy pattern.

```
SpotifyService.java X
15 import org.codehaus.jackson.JsonNode;
16 import org.codehaus.jackson.map.ObjectMapper;
17 import org.codehaus.jackson.node.ArrayNode;
18 import org.codehaus.jackson.node.ObjectNode;
19 import org.slf4j.Logger;
20 import org.slf4j.LoggerFactory;
21
22 import com.google.common.util.concurrent.AbstractIdleService;
23
24
25 public class SpotifyService extends AbstractIdleService implements OnlineServiceProviderStrategy{
26
28+     * Logger.□
30     private static final Logger log = LoggerFactory.getLogger(SpotifyService.class);
31     final String CLIENT_ID="385a2ee78cc14c8793cf29c08aa63f91";
32     final String CLIENT_SECRET="f3e4f9a58d2d4fd6a4a45dad6cfcd6be0";
33     final String AUTH_URL = "https://accounts.spotify.com/api/token";
34
36+     protected void startUp() throws Exception {□
41
43+     protected void shutDown() throws Exception {□
48
49+     private String getAccessToken() throws IOException {□
77
79+     public ArrayNode search(String query, String type) throws IOException {□
100
101+    private ArrayNode createPodcastObject(ArrayNode items) {□
115
116+    private ArrayNode createAudiobookObject(ArrayNode items) {□
132
```

ItunesService: corresponding to the concrete Strategy classes in the Strategy pattern.

```
1 package com.sismics.books.core.service;
2
3+import java.io.IOException;[]
20
21
22 public class ItunesService extends AbstractIdleService implements OnlineServiceProviderStrategy{
23
25+    * Logger.[]
27    private static final Logger log = LoggerFactory.getLogger(ItunesService.class);
28
29+    protected void startUp() throws Exception {[]
35
36+    protected void shutDown() throws Exception {[]
42
44+    public ArrayNode search(String query, String type) throws IOException {[]
63
64+    private ArrayNode createPodcastObject(ArrayNode items) {[]
78
79+    private ArrayNode createAudiobookObject(ArrayNode items) {[]
95
```

AudiobookPodcastService: corresponding to the Context class in Strategy pattern

```
1 package com.sismics.books.core.service;
2
3 import java.io.IOException;
4
5 public class AudiobookPodcastService {
6     private OnlineServiceProviderStrategy onlineServiceProviderStrategy;
7
8     public AudiobookPodcastService(OnlineServiceProviderStrategy onlineServiceProviderStrategy) {
9         this.onlineServiceProviderStrategy = onlineServiceProviderStrategy;
10    }
11    public ArrayNode search(String query, String type) throws IOException {
12        return this.onlineServiceProviderStrategy.search(query, type);
13    }
14
15 }
16 }
```

AudiobookPodcastResource: This is a controller class which is the first endpoint where the frontend makes connection with backend.

```
1 package com.sismics.books.rest.resource;
2
3 import java.io.IOException;
4
5 /**
6  * Tag REST resources.
7  *
8  * @author bgamard
9  */
10 @Path("/thirdparty")
11 public class AudiobookPodcastResource extends BaseResource{
12     public Response search()
13
14     private List<JSONObject> createJsonObject(ArrayNode items) throws JSONException {
15
16     public Response getFavouritePodcasts()
17
18     public Response getFavouriteAudiobooks()
19
20     public Response savePodcast()
21
22     public Response saveAudiobook()
23 }
```

Demo:

Search

Search Type

Audiobooks

Podcasts

Search On Spotify

Enter query

Fetching podcasts from Spotify:

Search

Search Type

Audiobooks

Podcasts

Search On Spotify

nikhil kamath



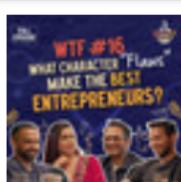
Zerodha's Nikhil Kamath On Dropping Out Of School To Become Successful | The Ranveer Show 94

▶ 0:00 / 0:47

🔇 ⏴

⋮

Heart icon



WTF Ep# 16 | What character 'flaws' make the best entrepreneurs? Nikhil ft. Ritesh, Ghazal, and Manish

▶ 0:00 / 1:00

🔇 ⏴

⋮

Heart icon



Dropout To Billionaire - Zerodha's Founder Nikhil Kamath's Journey To SUCCESS - FO5| Raj Shamani

▶ 0:00 / 1:00

🔇 ⏴

⋮

Heart icon



Ep# 15 | WTF is Climate Change? Nikhil ft. Sunita, Bhumi, Navroz and Mirik

▶ 0:00 / 1:00

🔇 ⏴

⋮

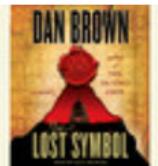
Heart icon

Fetching audiobook from Spotify:

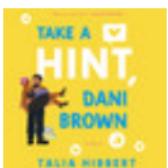
Search

dan brown

Search Type

 Audiobooks Podcasts**Search On Spotify****The Lost Symbol****Author:** Dan Brown**Narrator:** Paul Michael

Author(s): Dan Brown
 Narrator(s): Paul Michael
 In this stunning follow-up to the global phenomenon *The Da Vinci Code*, writer Dan Brown and narrator Paul Michael return to Washington, D.C., to solve another deadly mystery. *The Lost Symbol* is a masterstroke of storytelling—a deadly race through a real-world labyrinth of codes, secrets, chambers, tunnels, and temples of Washington, D.C., that accelerates through a startling landscape toward a single, inconceivable truth.

**Take a Hint, Dani Brown: A Novel****Author:** Talia Hibbert**Narrator:** Lone Butler

Author(s): Talia Hibbert
 Narrator(s): Lone Butler
 One of Oprah Magazine's best-selling authors, Talia Hibbert, has a new novel out. *Take a Hint, Dani Brown* follows Dani, a former sports star who is now a successful author, as she tries to navigate her way around the bedroom. She's determined to sleep together with her boyfriend, Zafir Ansari, but he's destined to sleep alone. Before long, she's tackling her fears into the dirt. But the former sports star has issues of her own, including a new romance with a brooding security guard named Zafir Ansari. Lying to help him children? Who on earth would refuse? Dani's plan is simple: fake a relationship with him to corrupt Dani's stone-cold realism. Before long, he's tackling her fears into the dirt. But the former sports star has issues of her own, including a new romance with a brooding security guard named Zafir Ansari.

**The Da Vinci Code: A Novel****Author:** Dan Brown**Narrator:** Paul Michael

Author(s): Dan Brown
 Narrator(s): Paul Michael
 While in Paris on business, Harvard symbologist Robert Langdon receives an e-mail from his old professor, Sienna Brooks, asking him to help solve a baffling cipher. While working to solve the enigmatic riddle, Langdon is stunned to discover it leads to a trail of clues hidden throughout the city.

Fetching podcasts from iTunes:

Search

nikhil kamath

Search Type

 Audiobooks Podcasts[Search On iTunes](#)

WTF Ep# 16 | What character 'flaws' make the best entrepreneurs? Nikhil ft. Ritesh, Ghazal, and Manish



Ep #11 | WTF Goes into Building a Fashion, Beauty, or Home Brand? Nikhil w/ Kishore, Raj, and Ananth



Ep# 13 | WTF does it take to Build Influence Today? Nikhil w/ Nuseir, Tanmay, Prajakta & Ranveer



Ep# 14 | WTF is Happening with EV? Nikhil ft. Founders of Reva, Ather, Blusmart, and Ossus



Fetching audiobooks from iTunes:

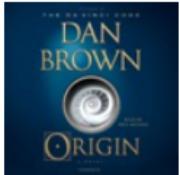
Search

dan brown

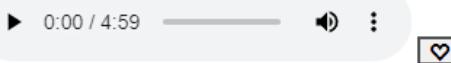
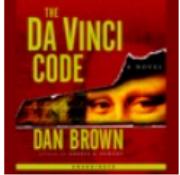
Search Type

 Audiobooks Podcasts

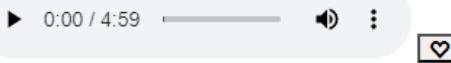
Search On iTunes

Origin: A Novel (Unabridged)**Author:** Dan Brown

The #1 New York Times Bestseller (October 2017) from the author of The Da Vinci Code.宗教图腾，到达超现代的毕尔巴鄂古根海姆博物馆，出席一个重大公告——揭幕。Kirsch，谁是其中之一的神学问题。事件开始时，Langdon和许多人都在。但精心策划的晚上突然爆发成混乱，Kirsch的宝贵发现使他成为Amber Vidal，博物馆馆长，与Kirsch合作策划这个挑衅性事件。他们一起逃到黑暗的历史长廊和极端宗教中，Langdon和Vidal必须避开一个受苦的敌人，他通过现代艺术和神秘符号，最终将他们引向Kirsch的最聪明、最有趣的书。

**The Da Vinci Code: A Novel (Unabridged)****Author:** Dan Brown

#1 WORLDWIDE BESTSELLER While in Paris, Harvard symbologist Robert Langdon is awakened by a phone call. Sophie Neveu sort through the bizarre riddles, they are stunned to discover a trail of clues hidden in the works of Leonardo da Vinci—a secret society whose members included Sir Isaac Newton, Victor Hugo, and Da Vinci—and his faceless adversary who shadows their every move—the explosive, ancient truth will be lost forever.

**The Lost Symbol (Unabridged)****Author:** Dan Brown

In this stunning follow-up to the global phenomenon **The Da Vinci Code**, Dan Brown demonstrates once again real-world labyrinth of codes, secrets, and unseen truths . . . all under the watchful eye of Brown's most terrifying villain to date. Landscape toward an unthinkable finale. As the story opens, Harvard symbologist Robert Langdon is summoned to turn. A disturbing object —artfully encoded with five symbols—is discovered in the Capitol Building. Langdon recognizes the beloved mentor, Peter Solomon—a prominent Mason and philanthropist—is brutally kidnapped. Langdon realizes his only hope world of Masonic secrets, hidden history, and never-before-seen locations—all of which seem to be dragging him toward a sin-

Adding audiobooks and podcasts in favourites:

Search

nikhil kamath

Search Type

 Audiobooks Podcasts

Search On iTunes



WTF Ep# 16 | What character 'flaws' make the best entrepreneurs? Nikhil ft. Ritesh, Ghazal, and Manish



Ep #11 | WTF Goes into Building a Fashion, Beauty, or Home Brand? Nikhil w/ Kishore, Raj, and Ananth



Ep# 13 | WTF does it take to Build Influence Today? Nikhil w/ Nuseir, Tanmay, Prajakta & Ranveer



Ep# 14 | WTF is Happening with EV? Nikhil ft. Founders of Reva, Ather, Blusmart, and Ossus



Ep# 12 | WTF is The Restaurant Game? Nikhil w/ Pooja Dhingra, Zorawar Kalra & Riyaaz Amlani



Ep #6 | WTF is Health? with Nikhil Kamath, Suniel Shetty, Nitin Kamath and Mukesh Bansal

Result after adding music in favourites section:

Choose Type

 Audiobooks Podcasts

Choose Provider

 Spotify iTunes Fetch

Ep# 12 | WTF is The Restaurant Game? Nikhil w/ Pooja Dhingra, Zorawar Kalra & Riyaaz Amlani

 0:00 / 3:23:53 — ⋮


WTF Ep# 16 | What character 'Flaws' make the best entrepreneurs? Nikhil ft. Ritesh, Ghazal, and Manish

 0:00 / 4:14:43 — ⋮


Ep# 13 | WTF does it take to Build Influence Today? Nikhil w/ Nuseir, Tanmay, Prajakta & Ranveer

 0:00 / 3:09:14 — ⋮

Contribution of team members :

1. Gunank Singh Jakhar -

- Online integration of Spotify and iTunes

2. M Elamparithy -

- Enhance user management system (Registration and Login)
- User self-registration

3. Rishabh Gupta -

- Book Ranking
- Book Filtering

4. Piyush Singh -

- Bonus Task (Public and Private Book self)

5. Prashant Kumar -

- Common Library Implementation(Adding a book, Common library display subsystem, Rating)