

Book.java

```
package com.lib.LiMS.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "books")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Book {
    @Id
    private String id;

    private String title;
    private String author;
    private String genre;
    private int publicationYear;
    private String shelfLocation;
}
```

BookRepository.java

```
package com.lib.LiMS.repository;

import com.lib.LiMS.model.Book;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

public interface BookRepository extends MongoRepository<Book, String> {

    List<Book> findByPublicationYear(int publicationYear);

    Optional<Book> findGenreById(String id);

    @Transactional
    void deleteByPublicationYear(int publicationYear);
}
```


BookService.java

```
package com.lib.LiMS.service;

import com.lib.LiMS.model.Book;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public interface BookService {
    Book createBook(Book book);
    List<Book> getAllBooks();
    Book getBookById(String id);
    Book updateBook(String id, Book book);
    void deleteBookById(String id);
    List<Book> getBooksByYear(int year);
    Optional<String> getGenreById(String id);
    void deleteByYear(int publicationYear);
}
```

BookServiceImpl.java

```
package com.lib.LiMS.service.impl;

import com.lib.LiMS.model.Book;
import com.lib.LiMS.repository.BookRepository;
import com.lib.LiMS.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class BookServiceImpl implements BookService {

    @Autowired
    private BookRepository bookRepository;

    @Override
    public Book createBook(Book book) {
        return bookRepository.save(book);
    }

    @Override
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }
}
```



```
@Override
public Book getBookById(String id) {
    Optional<Book> book = bookRepository.findById(id);
    return book.orElse(null);
    //return null or throw an exception
}

@Override
public Book updateBook(String id, Book book) {
    if (bookRepository.existsById(id)) {
        book.setId(id);
        return bookRepository.save(book);
    }
    return null;
}

@Override
public void deleteBookById(String id) {
    bookRepository.deleteById(id);
}

@Override
public List<Book> getBooksByYear(int year) {
    return bookRepository.findByPublicationYear(year);
}

@Override
public Optional<String> getGenreById(String id) {
    Optional<Book> book = bookRepository.findGenreById(id);
    return book.map(Book::getGenre);
}

@Override
public void deleteByYear(int publicationYear) {
    bookRepository.deleteByPublicationYear(publicationYear);
}
}
```

BookController.java

```
package com.lib.LiMS.controller;

import com.lib.LiMS.model.Book;
import com.lib.LiMS.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
```



```
import java.util.List;
import java.util.Optional;

@RestController
@CrossOrigin(origins = "*")
@RequestMapping("api/books")
public class BookController {

    @Autowired
    private BookService bookService;

    @PostMapping
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        Book newBook = bookService.createBook(book);
        return ResponseEntity.ok(newBook);
    }

    @GetMapping
    public ResponseEntity<List<Book>> getAllBooks() {
        List<Book> books = bookService.getAllBooks();
        return ResponseEntity.ok(books);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Book> getBookById(@PathVariable String id) {
        Book book = bookService.getBookById(id);
        return book != null ? ResponseEntity.ok(book) :
        ResponseEntity.notFound().build();
    }

    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable String id,
    @RequestBody Book book) {
        Book updatedBook = bookService.updateBook(id, book);
        return updatedBook != null ? ResponseEntity.ok(updatedBook) :
        ResponseEntity.notFound().build();
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Book> deleteBook(@PathVariable String id) {
        bookService.deleteBookById(id);
        return ResponseEntity.noContent().build();
    }

    @GetMapping("/year/{year}")
    public ResponseEntity<List<Book>> getBooksByYear(@PathVariable int year) {
        List<Book> books = bookService.getBooksByYear(year);
        return ResponseEntity.ok(books);
    }

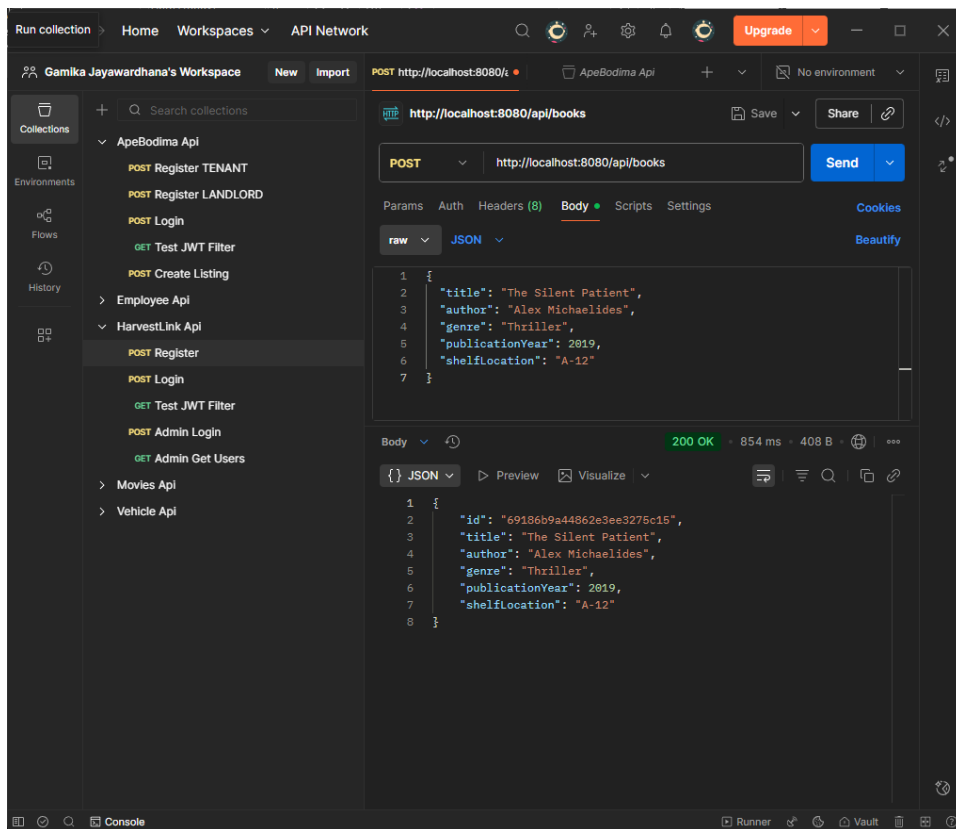
    @GetMapping("/{id}/genre")
    public ResponseEntity<String> getGenre(@PathVariable String id) {
        Optional<String> genre = bookService.getGenreById(id);
        return genre.map(ResponseEntity::ok).orElseGet(() ->
        ResponseEntity.notFound().build());
    }
}
```



```
}

    @DeleteMapping("/year/{year}")
    public ResponseEntity<String> deleteBookByYear(@PathVariable int year){
        bookService.deleteByYear(year);
        return ResponseEntity.ok("Successfully deleted all service record for
the year: " + year);
    }

}
```



LAB-06

EC/2021/006

W. K. G. K. JAYAWARDANA

Run collection Home Workspaces API Network

Gamika Jayawardhana's Workspace New Import

GET http://localhost:8080/api/books

Save Share

GET http://localhost:8080/api/books

Send

Params Auth Headers (8) Body Scripts Settings Cookies Beautify

raw JSON

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelflocation": "B-07"
7 }
8
```

Body JSON Preview Visualize

```
1 [
2   {
3     "id": "69186b9a44862e3ee3275c15",
4     "title": "The Silent Patient",
5     "author": "Alex Michaelides",
6     "genre": "Thriller",
7     "publicationYear": 2019,
8     "shelflocation": "A-12"
9   },
10  {
11    "id": "69186bd444862e3ee3275c16",
12    "title": "Clean Code",
13    "author": "Robert C. Martin",
14    "genre": "Programming",
15    "publicationYear": 2008,
16    "shelflocation": "B-07"
17  }
18 ]
```

200 OK · 279 ms · 561 B

Runner Vault

Run collection Home Workspaces API Network

Gamika Jayawardhana's Workspace New Import

PUT http://localhost:8080/api/books/69186b9a44862e3ee3275c15

Save Share

PUT http://localhost:8080/api/books/69186b9a44862e3ee3275c15

Send

Params Auth Headers (8) Body Scripts Settings Cookies Beautify

raw JSON

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelflocation": "B-07"
7 }
8
```

Body JSON Preview Visualize

```
1 {
2   "id": "69186b9a44862e3ee3275c15",
3   "title": "Clean Code",
4   "author": "Robert C. Martin",
5   "genre": "Programming",
6   "publicationYear": 2008,
7   "shelflocation": "B-07"
8 }
```

200 OK · 333 ms · 403 B

Runner Vault

LAB-06

EC/2021/006

W. K. G. K. JAYAWARDANA

The screenshot shows the Postman API client interface. The left sidebar displays a collection named 'ApeBodima Api' with several endpoints. The main panel shows a DELETE request to `http://localhost:8080/api/books/69186b9a44862e3ee...`. The 'Body' tab is selected, showing a JSON object:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelflocation": "B-07"
7 }
```

The response status is `204 No Content` with a response time of 396 ms and a body size of 201 B.

The screenshot shows the Postman API client interface. The left sidebar displays a collection named 'ApeBodima Api' with several endpoints. The main panel shows a GET request to `http://localhost:8080/api/books/year/2008`. The 'Body' tab is selected, showing a JSON object:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelflocation": "B-07"
7 }
```

The response status is `200 OK` with a response time of 299 ms and a body size of 405 B. The response body is shown in the 'JSON' tab:

```
1 [
2   {
3     "id": "69186bd444862e3ee3275c16",
4     "title": "Clean Code",
5     "author": "Robert C. Martin",
6     "genre": "Programming",
7     "publicationYear": 2008,
8     "shelflocation": "B-07"
9   }
10 ]
```


LAB-06

EC/2021/006

W. K. G. K. JAYAWARDANA

The screenshot shows the Postman interface with a workspace named "Gamika Jayawardhana's Workspace". The left sidebar lists collections, including "ApeBodima Api" and "HarvestLink Api". The main panel displays a GET request to `http://localhost:8080/api/books/69186bd444862...`. The response is a JSON object:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelfLocation": "B-07"
7 }
```

The status bar indicates a 200 OK response with a 356 ms latency and 264 B of data.

The screenshot shows the Postman interface with the same workspace. The main panel displays a DELETE request to `http://localhost:8080/api/books/year/2019`. The response is a JSON object:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "genre": "Programming",
5   "publicationYear": 2008,
6   "shelfLocation": "B-07"
7 }
```

The status bar indicates a 200 OK response with a 203 ms latency and 311 B of data. The response body is displayed as "1 Successfully deleted all service record for the year: 2019".