

## 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());
    }
}
```

## LAB-06

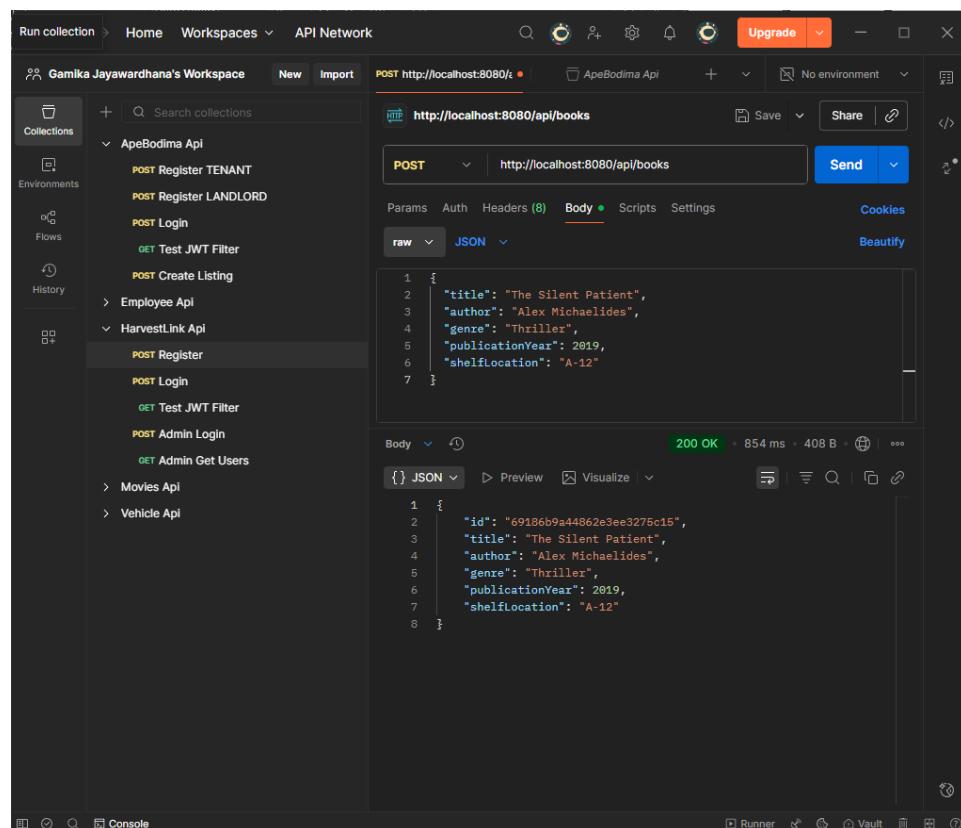
EC/2021/006

W. K. G. K. JAYAWARDANA

```
}
```

```
@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

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/books`. The response body is a JSON array containing two book objects:

```
[{"id": "69186b9a44862e3ee3275c15", "title": "The Silent Patient", "author": "Alex Michaelides", "genre": "Thriller", "publicationYear": 2019, "shelfLocation": "A-12"}, {"id": "69186bd444862e3ee3275c16", "title": "Clean Code", "author": "Robert C. Martin", "genre": "Programming", "publicationYear": 2008, "shelfLocation": "B-07"}]
```

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/books/69186b9a44862e3ee3275c15`. The response body is a JSON object representing the updated book record:

```
{"id": "69186b9a44862e3ee3275c15", "title": "Clean Code", "author": "Robert C. Martin", "genre": "Programming", "publicationYear": 2008, "shelfLocation": "B-07"}
```

# LAB-06

EC/2021/006

W. K. G. K. JAYAWARDANA

The screenshot shows the Postman application interface. On the left, the sidebar lists collections, environments, flows, and history. The main area shows a collection named "Gamika Jayawardhana's Workspace" with a sub-collection "ApeBodima Api". Under "ApeBodima Api", there are several POST requests: "Register TENANT", "Register LANDLORD", "Login", "Test JWT Filter", "Create Listing", "Employee Api", "HarvestLink Api", "Register", "Login", "Test JWT Filter", "Admin Login", and "Admin Get Users". A "DELETE" request is selected, targeting the URL <http://localhost:8080/api/books/69186b9a44862e3ee...>. The "Body" tab is active, showing a JSON payload:

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

The response status is 204 No Content, with a duration of 396 ms and a size of 201 B.

The screenshot shows the Postman application interface. The sidebar and collection structure are identical to the first screenshot. A "GET" request is selected, targeting the URL <http://localhost:8080/api/books/year/2008>. The "Body" tab is active, showing the same JSON payload as the previous screenshot:

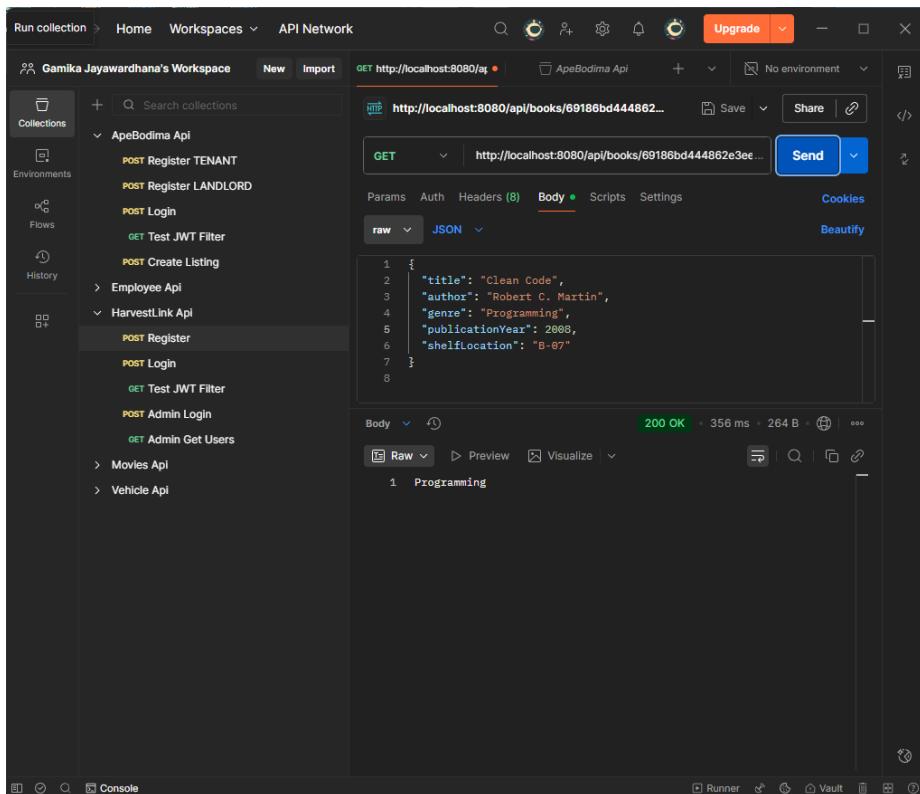
```
1: {  
2:   "title": "Clean Code",  
3:   "author": "Robert C. Martin",  
4:   "genre": "Programming",  
5:   "publicationYear": 2008,  
6:   "shelflocation": "B-07"  
7: }  
8:
```

The response status is 200 OK, with a duration of 299 ms and a size of 405 B.

# LAB-06

EC/2021/006

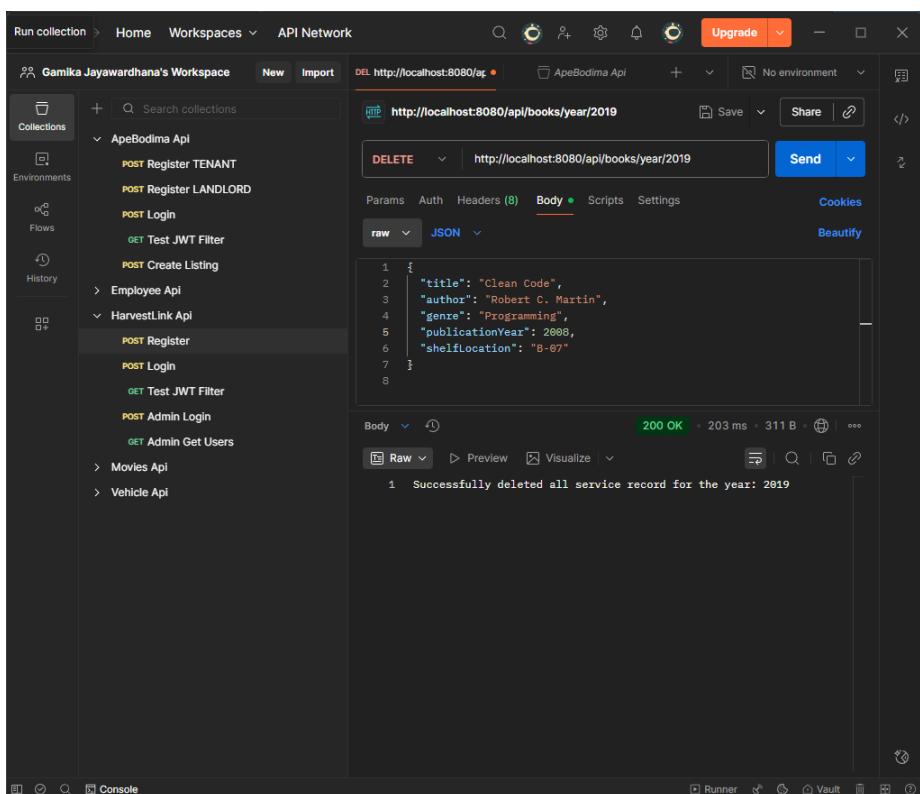
W. K. G. K. JAYAWARDANA



The screenshot shows the Postman interface with a collection named "Gamika Jayawardhana's Workspace". A specific API endpoint, "GET http://localhost:8080/api/books/69186bd444862e3ee...", is selected. The response body is displayed as JSON:

```
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 duration of 356 ms and a size of 264 B.



The screenshot shows the Postman interface with the same collection. A DELETE request is made to "http://localhost:8080/api/books/year/2019". The response body is displayed as JSON:

```
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 duration of 203 ms and a size of 311 B. The message in the body is: "Successfully deleted all service record for the year: 2019".