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.

PROMINEO TECH

Screenshots of Code:

```java
package com.promineotech.bookTracker;

import org.springframework.boot.SpringApplication;

@ComponentScan("com.promineotech.bookTracker")
@SpringBootApplication
public class App
{
    public static void main( String[] args )
    {
        SpringApplication.run(App.class, args);
    }
}
```

```java
package com.promineotech.bookTracker.controller;

import org.springframework.beans.factory.annotation.Autowired;
//import com.promineotech.bookTracker.service.BookService;

@RestController
@RequestMapping("users/{id}/authors")
public class AuthorController {

    @Autowired
    private AuthorService service;

//  private BookService bookService;

    @RequestMapping(value = "/{authorId}", method = RequestMethod.GET)
    public ResponseEntity<Object> getAuthor(@PathVariable Long authorId) {
        try {
            return new ResponseEntity<Object>(service.getAuthorById(authorId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(method = RequestMethod.GET)
    public ResponseEntity<Object> getAuthors() {
        return new ResponseEntity<Object>(service.getAuthors(), HttpStatus.OK);
    }

//  @RequestMapping(value = "/{authorId}/books")
//  public ResponseEntity<Object> getBooks(){
//      return new ResponseEntity<Object>(bookService.getBooks(), HttpStatus.OK);
//  }

    @RequestMapping(method = RequestMethod.POST)
    public ResponseEntity<Object> addAuthor(@RequestBody Author author) {
        return new ResponseEntity<Object>(service.addAuthor(author), HttpStatus.CREATED);
    }

    @RequestMapping(value = "/{authorId}", method = RequestMethod.PUT)
    public ResponseEntity<Object> updateAuthor(@RequestBody Author author, @PathVariable Long authorId) {
        try {
            return new ResponseEntity<Object>(service.updateAuthor(author, authorId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value = "/{authorId}", method = RequestMethod.DELETE)
    public ResponseEntity<Object> removeAuthor(@PathVariable Long authorId) {
        try {
            service.removeAuthor(authorId);
            return new ResponseEntity<Object>("Successfully removed author with id: " + authorId, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }
```

PROMINEO TECH

```java
package com.promineotech.bookTracker.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("users/{id}/books")
public class BookController {

    @Autowired
    private BookService service;

    @RequestMapping(value="/{bookId}", method=RequestMethod.GET)
    public ResponseEntity<Object> getBook(@RequestBody Long bookId) {
        try {
            return new ResponseEntity<Object>(service.getBookById(bookId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getBooks() {
        return new ResponseEntity<Object>(service.getBooks(), HttpStatus.OK);
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> addBook(@RequestBody Book book, @PathVariable Long id) throws Exception {
        return new ResponseEntity<Object>(service.addBook(book, id), HttpStatus.CREATED);
    }

    @RequestMapping(value="/{bookId}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateBook(@RequestBody Book book, @PathVariable Long bookId) {
        try {
            return new ResponseEntity<Object>(service.updateBook(book, bookId), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value="/{bookId}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> removeBook(@PathVariable Long id) {
        try {
            service.removeBook(id);
            return new ResponseEntity<Object>("Successfully removed book with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }
}
```

PROMINEO TECH

```java
package com.promineotech.bookTracker.controller;

import java.nio.file.Files;

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

    private static String UPLOADED_FOLDER = "./pictures/";

    @Autowired
    private UserService service;

    @RequestMapping(value = "/register", method = RequestMethod.POST)
    public ResponseEntity<Object> register(@RequestBody User user) {
        return new ResponseEntity<Object>(service.createUser(user), HttpStatus.CREATED);
    }

    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public ResponseEntity<Object> login(@RequestBody User user) {
        try {
            return new ResponseEntity<Object>(service.login(user), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(method = RequestMethod.GET)
    public ResponseEntity<Object> getUsers() {
        return new ResponseEntity<Object>(service.getUsers(), HttpStatus.OK);
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getUser(@PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.getUserById(id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value = "/{id}/profilePicture", method = RequestMethod.POST)
    public ResponseEntity<Object> singleFileUpload(@PathVariable Long id, @RequestParam("file") MultipartFile file) {
        if (file.isEmpty()) {
            return new ResponseEntity<Object>("Please upload a file.", HttpStatus.BAD_GATEWAY);
        }

        try {
            String url = UPLOADED_FOLDER + file.getOriginalFilename();
            byte[] bytes = file.getBytes();
            Path path = Paths.get(url);
            Files.write(path, bytes);
            return new ResponseEntity<Object>(service.updateProfilePicture(id, url), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
        }
```

PROMINEO TECH

```java
@Entity
public class Author {

    private Long authorId;
    private String firstName;
    private String lastName;

    @JsonIgnore
    private Set<Book> books;

    @JsonIgnore
    private User user;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getAuthorId() {
        return authorId;
    }

    public void setAuthorId(Long authorId) {
        this.authorId = authorId;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @OneToMany(mappedBy = "author")
    public Set<Book> getBooks() {
        return books;
    }

    public void setBooks(Set<Book> books) {
        this.books = books;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
```

PROMINEO TECH

```java
package com.promineotech.bookTracker.entity;

import javax.persistence.Entity;

@Entity
public class Book {

    private Long bookId;
    private String title;
    private String genre = "Thriller";
    private Long publishYear;
    private String readDate;

    @Enumerated(EnumType.STRING)
    private BookRating rating = BookRating.GD;

    private Author author;

    private User user;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getBookId() {
        return bookId;
    }

    public void setBookId(Long bookId) {
        this.bookId = bookId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public Long getPublishYear() {
        return publishYear;
    }

    public void setPublishYear(Long publishYear) {
        this.publishYear = publishYear;
    }

```

```java
    public String getReadDate() {
        return readDate;
    }

    public void setReadDate(String readDate) {
        this.readDate = readDate;
    }

    public BookRating getRating() {
        return rating;
    }

    public void setRating(BookRating rating) {
        this.rating = rating;
    }

    @ManyToOne
    @JoinColumn(name = "authorId")
    public Author getAuthor() {
        return author;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}

```

```java
package com.promineotech.bookTracker.repository;

import org.springframework.data.repository.CrudRepository;

public interface AuthorRepository extends CrudRepository<Author, Long> {

}

```

```java
package com.promineotech.bookTracker.repository;

import org.springframework.data.repository.CrudRepository;

public interface BookRepository extends CrudRepository<Book, Long> {

}

```

```java
package com.promineotech.bookTracker.repository;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

    public User findByUsername(String username);
}

```

PROMINEO TECH

```java
package com.promineotech.bookTracker.service;

import org.apache.logging.log4j.LogManager;

@Service
public class AuthorService {

    private static final Logger logger = LogManager.getLogger(AuthorService.class);

    @Autowired
    private AuthorRepository repo;

    public Author getAuthorById(Long id) throws Exception {
        try {
            return repo.findOne(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to retrieve Author.");
            throw e;
        }
    }

    public Iterable<Author> getAuthors() {
        return repo.findAll();
    }

    public Author addAuthor(Author author) {
        return repo.save(author);
    }

    public Author updateAuthor(Author author, Long id) throws Exception {
        try {
            Author oldAuthor = repo.findOne(id);
            oldAuthor.setFirstName(author.getFirstName());
            oldAuthor.setLastName(author.getLastName());
            oldAuthor.setBooks(author.getBooks());
            return repo.save(oldAuthor);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to update author.");
            throw new Exception("Unable to update author.");
        }
    }

    public void removeAuthor(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to delete author.");
            throw new Exception("Unable to remove author.");
        }
    }
}
```

PROMINEO TECH

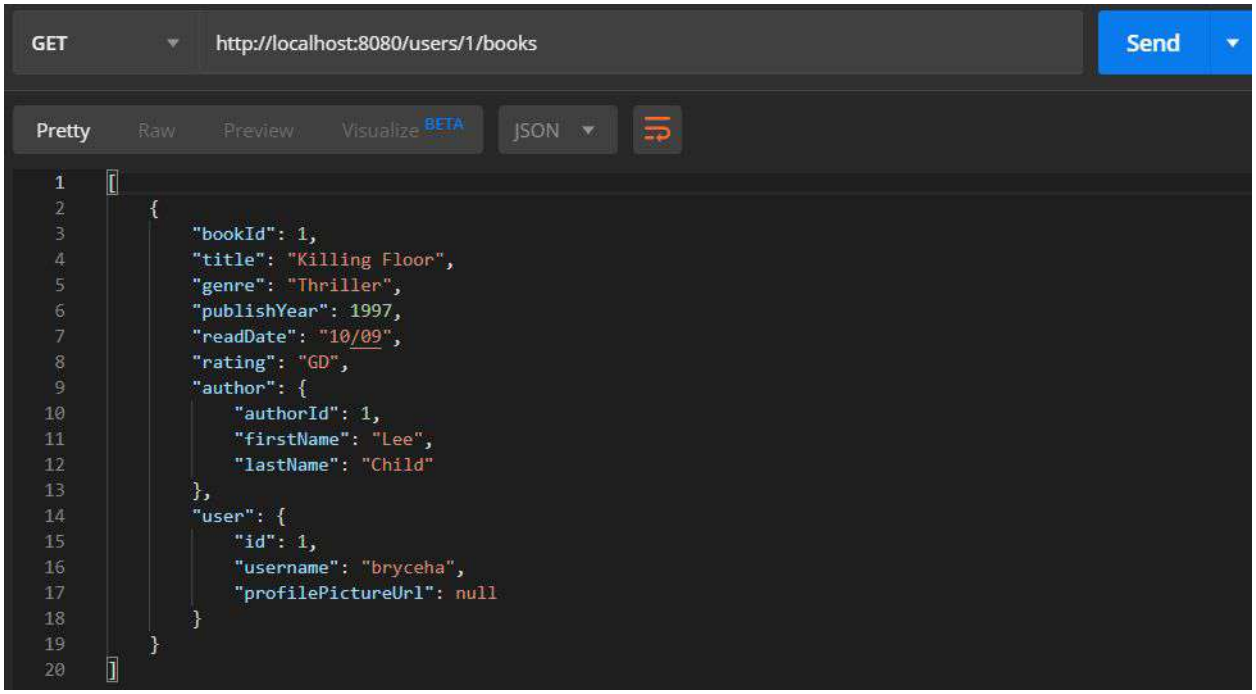
```java
@Service
public class BookService {

    private static final Logger logger = LogManager.getLogger(BookService.class);

    @Autowired
    private BookRepository repo;

    public Iterable<Book> getBooks() {
        return repo.findAll();
    }

    public Book getBookById(Long id) throws Exception {
        try {
            return repo.findOne(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to retrieve book.");
            throw e;
        }
    }

    @Autowired
    private AuthorRepository authorRepo;

    public Book addBook(Book book, Long userId) throws Exception {
        Author author = authorRepo.findOne(userId);
        if (author == null) {
            logger.error("Exception occured while trying to add book.");
            throw new Exception("Author does not exist in database.");
        }
        book.setAuthor(author);
        return repo.save(book);
    }

    public Book updateBook(Book book, Long id) throws Exception {
        try {
            Book oldBook = repo.findOne(id);
            oldBook.setAuthor(book.getAuthor());
            oldBook.setGenre(book.getGenre());
            oldBook.setPublishYear(book.getPublishYear());
            oldBook.setRating(book.getRating());
            oldBook.setReadDate(book.getReadDate());
            oldBook.setTitle(book.getTitle());
            return repo.save(oldBook);
        } catch (Exception e) {
            logger.error("Exception occured while trying to update book.");
            throw new Exception("Unable to update book.");
        }
    }

    public void removeBook(Long id) throws Exception {
        try {
        repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to delete book: " + id, e);
            throw new Exception("Unable to remove book.");
```

PROMINEO TECH

```java
package com.promineotech.bookTracker.service;

import org.apache.logging.log4j.LogManager;

@Service
public class UserService {

    private static final Logger logger = LogManager.getLogger(UserService.class);

    @Autowired
    private UserRepository repo;

    public User createUser(User user) {
        return repo.save(user);
    }

    public User login(User user) throws Exception {
        User foundUser = repo.findByUsername(user.getUsername());
        if (foundUser != null && foundUser.getPassword().equals(user.getPassword())) {
            return foundUser;
        } else {
            throw new Exception("Invalid username or password.");
        }
    }

    public User getUserById(Long id) throws Exception {
        try {
            return repo.findOne(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to retrieve user: " + id, e);
            throw e;
        }
    }

    public Iterable<User> getUsers() {
        return repo.findAll();
    }

    public User updateProfilePicture(Long userId, String url) throws Exception {
        User user = repo.findOne(userId);
        if (user == null) {
            throw new Exception("User does not exist.");
        }
        user.setProfilePictureUrl(url);
        return repo.save(user);
    }
}

```

```java
package com.promineotech.bookTracker.util;

public enum BookRating {

    GD,
    OK,
    NG;

}

```

Screenshots of Running Application:



URL to GitHub Repository:

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