03. Books

You can check your solutions in Judge.

Your Task

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **`bookService`**, which represents an object. You may use the following code as a template:

```
describe("Tests …", function() {
    describe("TODO …", function() {
        it("TODO …", function() {
            // TODO: …
        });
     });
     // TODO: …
});
```

The **`bookService`** object that should have the following functionality:

- **`getBooks()`** - A function that returns an object with a response from a book service.
    - If the request is successful, return an object with status **200** and an array of **3 books**, each containing **id**, **title**, **author**, **year** and **genre**.
- **`addBook(book)`** - A function that adds a new book to the list of books.
    - **book** is an object that must contain the fields **id**, **title**, **author**, **year** and **genre**.
    - If the book is successfully added, return an object with status **201** and a success message: "**Book added successfully**."
    - If any of the fields are missing or invalid, return an object with status **400** and an error message: "**Invalid Book Data**!"
- **`deleteBook(bookID)`** - A function that deletes a book by its id.
    - **`bookID`** is a string with the Id of the book to be deleted
    - If the book is successfully deleted, return an object with status **200** and a success message: "**Book deleted successfully**."
    - If the book is not found, return an object with status **404** and an error message: "**Book Not Found**!"
- **`updateBook(oldId, newBook)`** - A function that updates the information of an existing book.
    - **`oldId`** is a string with the id of the book to be updated.
    - **newBook** is an object containing the new book data.
    - If the book is successfully updated, return an object with status **200** and a success message: "**Book updated successfully**."
    - If the book is **not found**, return an object with status **404** and an error message: "**Book Not Found**!"
    - If any property of the new book is **empty or missing**, return a **400** status and an error message: "**Invalid Book Data!**"

JS Code

To ease you in the process, you are provided with an implementation that meets all of the specification requirements for the **`bookService`** object:

bookService.js

```
const bookService = {
    books: [
        { id: "1", title: "1984", author: "George Orwell", year: 1949, genre: "Dystopian" },
        { id: "2", title: "To Kill a Mockingbird", author: "Harper Lee", year: 1960, genre: "Fiction" },
        { id: "3", title: "The Great Gatsby", author: "F. Scott Fitzgerald", year: 1925, genre: "Classic" }
    ],

    // getBooks() - Returns a list of books with status 200
    getBooks() {
        return {
            status: 200,
            data: this.books
        };
    },

    // addBook(book) - Adds a new book to the list. If any field is missing or invalid, returns status 400
    addBook(book) {
        if (!book.id || !book.title || !book.author || !book.year || !book.genre) {
            return {
                status: 400,
                error: "Invalid Book Data!"
            };
        }
        this.books.push(book);
        return {
            status: 201,
            message: "Book added successfully."
```

```
        };
    },

    // deleteBook(bookID) - Deletes a book by ID. If not found, returns status 404
    deleteBook(bookID) {
        const bookIndex = this.books.findIndex(book => book.id === bookID);
        if (bookIndex === -1) {
            return {
                status: 404,
                error: "Book Not Found!"
            };
        }
        this.books.splice(bookIndex, 1);
        return {
            status: 200,
            message: "Book deleted successfully."
        };
    },

    // updateBook(oldId, newBook) - Updates an existing book by ID. If not found,
returns status 404
    // If new book data is invalid, returns status 400
    updateBook(oldId, newBook) {
        const bookIndex = this.books.findIndex(book => book.id === oldId);
        if (bookIndex === -1) {
            return {
                status: 404,
                error: "Book Not Found!"
            };
        }
        if (!newBook.id || !newBook.title || !newBook.author || !newBook.year ||
!newBook.genre) {
            return {
                status: 400,
                error: "Invalid Book Data!"
            };
```

```
        }
        this.books[bookIndex] = newBook;
        return {
            status: 200,
            message: "Book updated successfully."
        };
    }
};
```

Submission

Submit your tests inside a **`describe()`** statement, as shown above.