Assignment 11

Using mongoose defines the following collections (10 Grades):

Users (0.5 Grade)	Books (0.5 Grade)	Library (0.5 Grade)	BorrowedBook(0.5Grade)
• **name** (String, required) • **email** (String, Unique, required) • **Password** (String, required) • **Phone** (String, required) • **borrowedBooks** (array of booksId) (ref books)	• **title** (String, required) • **author** (ref to Users, required) • **publishedYear** (Number, required) • **genre** (String, required) • **availableCopies** (Number, required)	• **name** (String, required) • **location** (String, required) • **books** (array of booksId)	• **userId** (ref user, required) • **bookId** (ref book, required) • **borrowedAt** (timestamp, required) • **dueDate** (timestamp, required) • **returned** (Boolean)

Using graphQl implements the following tasks:

A- Mutations (6 Grades)

1. **Signup (make sure that the email does not exist before) (Don't forget to hash the password and encrypt the phone). (1 Grade)**

```
mutation {
  registerUser(input: { name: "Alice Doe", email: "alice@example.com", password: "securePass123" }) {
    id
    email
  }
}
```

```
{
  "data": {
    "registerUser": {
      "id": "64d91c42d8979e1f30a12345",
      "email": "alice@example.com"
    }
  }
}
```

2. **Login user and return JWT token. (1 Grade)**

```
mutation {
  loginUser(email: "alice@example.com", password: "securePass123") {
    token
  }
}
```

```
{
  "data": {
    "loginUser": {
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
    }
  }
}
```

3. **Add a new book. (1 Grade)**

```
mutation {
  addBook(input: { title: "New Book", author: "Jane Doe", publishedYear: 2023, genre: "Fiction", availableCopies: 10 }) {
    id
    title
  }
}
```

```
{
  "data": {
    "addBook": {
      "id": "2",
      "title": "New Book"
    }
  }
}
```

4. **Borrow book (for 2 days). (1 Grade)**

```
mutation {
  borrowBook(bookId: "1") {
    id
    title
    availableCopies
  }
}
```

```
{
  "data": {
    "borrowBook": {
      "id": "1",
      "title": "GraphQL Basics",
      "availableCopies": 4
    }
  }
}
```

5. **Delete a user (authenticated users only). (1 Grade)**

```
mutation {
  deleteUser {
    message
  }
}
```

```
{
  "data": {
    "deleteUser": {
      "message": "User deleted successfully"
    }
  }
}
```

6. **Mark a book as available again (1 Grade)**

```
mutation {
  returnBook(bookId: "456") {
    message
  }
}
```

```
{
  "data": {
    "makeBookAvailable": {
      "message": "Book is now available for borrowing"
    }
  }
}
```

B- Queries (2 Grades)

7. **Retrieve all books. (0.5 Grade)**

```
query {
  getAllBooks {
    id
    title
    author
    publishedYear
    genre
    availableCopies
  }
}
```

```
{
  "data": {
    "getAllBooks": [
      {
        "id": "1",
        "title": "GraphQL Basics",
        "author": "John Smith",
        "publishedYear": 2020,
        "genre": "Programming",
        "availableCopies": 5
      }
    ]
  }
}
```

Assignment 11

8. **Retrieve Book by id. (0.5 Grade)**

```
query {
  getBookById(id: "1") {
    title
    author
    publishedYear
  }
}
```

```
{
  "data": {
    "getBookById": {
      "title": "GraphQL Basics",
      "author": "John Smith",
      "publishedYear": 2020
    }
  }
}
```

9. **Fetch libraries along with the books they contain. (0.5 Grade)**

```
query {
  getLibraries {
    id
    name
    books {
      title
      author
    }
  }
}
```

```
{
  "data": {
    "getLibraries": [
      {
        "id": "1",
        "name": "Central Library",
        "books": [
          {
            "title": "GraphQL Basics",
            "author": "John Smith"
          }
        ]
      }
    ]
  }
}
```

10. **Retrieve overdue borrowed books that have not been returned. (0.5 Grade)**

```
query {
  getOverdueBooks {
    id
    userId
    bookId
    dueDate
  }
}
```

```
{
  "data": {
    "getOverdueBooks": [
      {
        "id": "1",
        "userId": "123",
        "bookId": "456",
        "dueDate": "2024-06-10"
      }
    ]
  }
}
```

⚠️ Important Notes about postman

1. **Name the endpoint with a meaningful name like 'Add User', not dummy names.**
2. **Save your changes on each request (ctrl+s).**
3. **Include the Postman collection link (export your Postman collection) in the email with your assignment link**

Bonus (2 Grades)

How to deliver the bonus?

1- Solve the problem Is Subsequence on **LeetCode**

2- Inside your assignment folder, create a **SEPARATE FILE** and name it “bonus.js”

3- Copy the code that you have submitted on the website inside ”bonus.js” file