**LAB ASSIGNMENT 6**

**Name: Sanket Hol**

**PRN:12310611**

**Class: TY AIML-A**

**Batch: 03**

**Roll No: 44**

**Experiment Number: 6**

**Title: Design and develop a responsive website for an Online Book Store using REACT, Node JS / PHP and MySQL / MongoDB / Oracle having:**

1. Home Page 2) Login Page 3) Catalogue Page 4) Registration Page (Database)

**Theory:**

Understanding Client–Server Architecture using modern web development technologies is essential for building scalable applications. This experiment demonstrates how to design and develop a three-tier Online Book Store Web Application using:

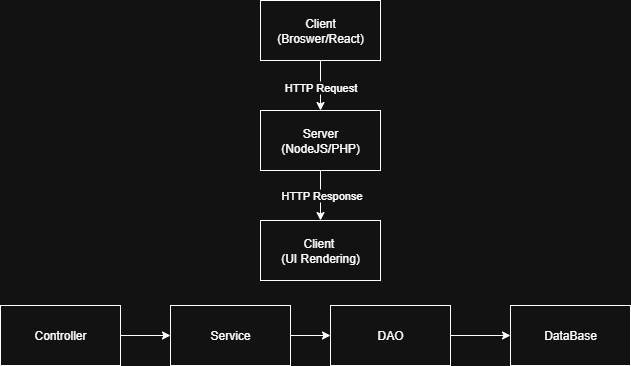
Front-end: React JS

Back-end: Node JS / Express (or PHP)

Database: MySQL / MongoDB / Oracle

Communication: REST API (GET, POST, PUT, DELETE)

**Client–Server Architecture Layers:**



**Pages Required**

1. **Home Page** – Featured books, categories, search bar, responsive UI.
2. **Login Page** – User authentication using Email & Password.
3. **Catalogue Page** – Books fetched from database via REST API, filterable by category/author.
4. **Registration Page** – User signup stored in database.

Each page must be responsive, optimized for desktop + mobile.

**Database Overview**

1. **Users Table**

user\_id (PK)

name

email

password\_hash

phone

created\_at

1. **Books Table**

book\_id (PK)

title

author

category

price

stock

description

cover\_url

The Registration Page saves records to the Users table through an API endpoint such as:

POST /api/auth/register

**REST API Overview**

| **METHOD** | **ENDPOINT** | **PURPOSE** |
| --- | --- | --- |
| GET | /api/books | Fetch all books |
| GET | /api/books/:id | Fetch single book |
| POST | /api/auth/register | Create user account |
| POST | /api/auth/login | Authenticate user |
| POST | /api/books | Add book (admin) |
| PUT | /api/books/:id | Update book |
| DELETE | /api/books/:id | Delete book |

**Procedure:**

* Create a project using spring boot initialize website
* Add Dependencies: Spring Web.......MySQL Driver......Spring Data JPA
* Click on: Generate .................it will generate jar file in download folder
* Extract it
* Open it in ID
* Import - as a maven project
* Tick on ------ pom.xml
* It will update
* Let us build controller : Controller for REST API (REST - Representational State Transfer)
* Create entity class
* Generate constructors, getters and setters for entity class
* Create interface class
* Create Service Class
* Database Connectivity: using JPA
* Extends it to JPA Repository:
* Service Implementation:
* Use – React JS to create front end and axios for routing

**A. Backend Development (NodeJS + Express)**

1. Initialize Project

npm init -y

npm install express mysql2 cors bcrypt jsonwebtoken

1. Create Folder Structure

backend/

server.js

routes/

controllers/

services/

config/

models/

1. Database Connectivity

* Create a MySQL database: bookstore\_db
* Connect using mysql2 inside **config/db.js**

1. Build Controller for REST API

* authController.js → register, login
* bookController.js → list, add, delete books

1. Create Models

* User.js
* Book.js

Each model contains SQL queries (or ORM schema in MongoDB/Oracle)

1. Enable CORS + JSON Parsing

app.use(cors());

app.use(express.json());

**B. Frontend Development (React JS)**

1. Initialize Project

npx create-react-app frontend

2. Create Components

src/

components/

Header.js

Footer.js

BookCard.js

pages/

Home.js

Login.js

Register.js

Catalogue.js

3. Connect Frontend to Backend

Use Axios:

npm install axios

api.js

export default axios.create({

baseURL: "http://localhost:5000/api"

});

1. **Implement Pages**

**Home Page**

* Search bar
* Featured books section
* Responsive grid using Bootstrap/Tailwind

**Login Page**

* Email + Password fields
* API: POST /auth/login
* Store token/localStorage

**Catalogue Page**

* Fetch all books via /books API
* Card layout
* Filters (categories, authors, price)

**Registration Page**

* Form with:
  + Name
  + Email
  + Password
  + Phone
* On submit → POST /auth/register
* Show confirmation message

**C. Integration**

* Ensure routes are correct
* Test full workflow:
  + Register → Login → View Books

**Conclusion:**

The Online Book Store project demonstrates the complete workflow of a modern three-tier web application. By integrating React (Frontend), NodeJS/PHP (Backend) and MySQL/MongoDB/Oracle (Database), students learn:

* Client-side UI development
* REST API design
* Database CRUD operations
* End-to-end system integration
* Responsive web design

This experiment solidifies concepts of full-stack application development and real-world engineering practices.