FPT ACADEMY INTERNATIONAL

FPT – APTECH COMPUTER EDUCATION

Bookie Boo

Supervisor.	Lê Mộng Thúy	
Batch.	T1.2406.E1	
Group.	3	
Serial No.	Student Name	Enrollment Number
1	VŨ THỊ HOÀI THU	T1S2404004
2	THÂN THẾ LỘC	T1S2404009
3	NGUYỄN KHẢI NGUYÊN	T1S2405014
4	HỒ ANH TOÀN	....

Month 05 Year 2025

Design Plan:	Document Name: Problem Definition Document	
Effective Date:	Version : 1	Page No.:

Problem Definition of Bookibooo Online Bookstore

➢ **Introduction:**

In today's digital age, parents and caregivers are increasingly seeking convenient and reliable online platforms to access high-quality children's books that support their child's growth and development. However, finding a specialized website that provides age-appropriate, development-focused book selections remains a challenge.

➢ **Target_Users:**

1. The primary users of the Bookibooo online bookstore are:
2. Parents of young children (aged 0–6+)
3. Caregivers (grandparents, babysitters, nannies)
4. Teachers and educators in early childhood education
5. Bookstore admins and staff
6. Children (with parental supervision)

➢ **Background:**

The demand for high-quality children's books is increasing, especially books that help develop creativity, thinking, and observation skills for different age groups.

➢ **Current_Problems:**

Lack of a specialized, user-friendly online platform for parents to easily find and purchase age-appropriate and development-focused books for their children.

➢ **Objectives:**

To develop an online bookstore that provides a simple, effective, and child-friendly interface for users to search, select, and purchase books by age and developmental needs.

➢ **Problem_Statement:**

The absence of a focused and user-friendly online bookstore makes it difficult for parents and caregivers to find the best books for their children's development. Bookibooo aims to address this need.

➢ **Expected_Benefits:**

- Save time for parents and caregivers
- Support children's learning and development
- Provide an efficient management system for admins
- Build a community of book lovers

	Prepared By (Student)	Approved By (Teacher)
Date	Group 3	Lê Mộng Thúy

Design Plan:	Document Name: CRS	SWD/Form No. 2
Effective Date:	Version: 1	Page No.

1. List of Inputs to the System
User Inputs:

- Registration and login information (email, password, full name, user role).
- Filtering and searching books (by category, price, status: in stock, out of stock).
- Selecting books, choosing quantity, adding to cart, removing from cart.
- Checkout information (shipping address, phone number, payment method).
- Monthly book event participation or feedback.

Admin Inputs:

- Adding/editing/deleting books (title, author, price, description, image, age group, category).
- Managing orders (confirming, updating order status).
- Managing user accounts.
- Adding news, events, and promotional content.

System Inputs:

- Book database imports (uploading book lists via CSV, Excel, etc.).
- Book cover images uploaded from the local machine.

2. List of Outputs Expected from the System

- Display lists of books by age group, category, featured, and promotional offers.
- Search results with filter and sorting options.
- Book detail page (information, image, reviews).
- Shopping cart and user order history.
- Order confirmation notifications and shipping status updates.
- Admin dashboard (books, orders, users management).
- Revenue reports, best-selling book statistics (for admin).
- Event and news pages.
- Responsive interface for mobile and desktop devices.

3. Overview of Processes Involved in the System
Customer Processes:

- Register/Login to their account.
- Browse, search, and filter books; view book details.
- Add books to cart.
- Place orders by entering shipping details and choosing a payment method.

Admin Processes:

- Login to the admin dashboard.
- Manage book inventory (add, edit, delete books).
- Manage customer accounts.

- Manage and process orders (confirm, update, cancel).
- Manage content (news, events, banners, promotions).
- View reports and analytics.

4. Hardware and Software Required for Implementing the Project

Hardware Requirements:
- Developer PC/Laptop (8GB RAM, SSD, Windows).
- Stable Internet connection.

Software Requirements:
- **Frontend:**
  - ReactJS (JavaScript), CSS
  - Web browsers (Chrome)
- **Backend:**
  - Core PHP
  - Apache web server, XAMPP (for local development)
- **Database:**
  - MySQL
- **Others:**
  - Git (source code management)
  - VS Code (code editor)
  - Payment gateway sandbox (VNPay for payment integration)

	Prepared By (Student)	Approved By (Teacher)
Date		

Design Plan:	Document Name: CRS/ Customer Acceptance Criteria	SWD/Form No. 2
Effective Date:	Version: 1	Page No.

S.No	Customer's Acceptance Criteria
1.	User registration and login work securely with email validation.
2.	All book listings are displayed correctly by age group and category, with filtering and search functionality.
3.	Users can add books to the cart, update quantities, and successfully place orders.
4.	Admin dashboard allows management of books, users, and orders.
5.	The system supports responsive design for both desktop and mobile devices.
6.	All payments through the integrated payment gateway are processed and reflected in the order history.
7.	The system is secure and protected against common web vulnerabilities
8.	
9.	
10.	

Date	Prepared By (Student)	Approved By (Teacher)
	Group 3	Lê Mộng Thúy

Project Plan

1. Project Details

> **Name_of_the_Client:**
> *Bookiboo store*

Date_of_Project_Plan:
01-05-2025

Project_Vision/Objectives:
To develop an online children's bookstore that enables parents and caregivers to easily find, select, and purchase age-appropriate, development-focused books with a user-friendly and secure web interface.

Scope:
The project will deliver an e-commerce website for book sales, covering:

- All locations with Internet access (web-based platform)
- Key business processes: User registration, book catalog browsing, search and filter, cart management, order placement, payment integration, and administration functions.
- Services: Book listing, user account management, order management, basic statistics/reporting for admin.

Our_Understanding_of_the_Client_Organization:
Bookibooo specializes in providing high-quality, carefully curated children's books. The client requires a modern e-commerce platform for online sales, customer management, order processing, and basic analytics, with a simple admin panel for efficient management.

> **Project Organization with Responsibilities and Authorities:**

Name	Role	Responsibilities
[Vũ Thị Hoài Thu]	Project Manager	Overall project coordination, communication
[Thân Thế Lộc]	Backend Developer	Database design, API development, security
[Nguyễn Khải Nguyên]	Frontend Developer	UI/UX design, React implementation
[Vũ Thị Hoài Thu]	QA/Tester	Testing, bug reporting, user documentation
[Vũ Thị Hoài Thu]	Deployment/Support	Server setup, deployment, maintenance

2. Project Initiation/Requirement Documents

> Information required from the client:

- List and categories of books to be sold (title, author, age group, category, price, images, descriptions)
- Logo and branding materials

- Details of desired payment methods (e.g., VNPay, Cash on Delivery)
- Contact information for notifications/emails
- User roles and permission levels (if any customizations are needed)

Any preferred workflow or admin processes

3. Deliverables

- Customer Requirement Specification (CRS)
- System Design Document (Architecture, ERD, API specs)
- Installation Manual
- User Manual (for both customers and admin)
- Maintenance Manual
- Source code and code documentation

4. Project Dependencies

- Timely provision of book data and branding materials from the client
- Access to client server or hosting environment for deployment
- Payment gateway account activation (e.g., VNPay Sandbox or production keys)
- Domain name registration and DNS setup

Client feedback for UI/UX designs

5. Major Project Milestones

1. **Database and System Design Completion** – [01/05-07-05]
2. **Backend API Implementation** – [07/05-23/05]
3. **Frontend Development Completion** – [10/05-25-05]
4. **Integration Testing and QA** – [25/05-28/05]
5. **User Acceptance Testing (UAT)** – [25/05-28/05]
6. **Production Deployment** – [27/05-30/05]
7. **Project Closure/Sign-off** – [30/05]

6. Quality Plan

Review Activities:

- Weekly review meetings with the project team
- Monthly progress meeting with the client
- Peer code review for each major component
- Documentation review before final delivery

Testing Activities:

- Functional testing (feature-by-feature)
- Integration testing (frontend-backend interaction)
- Security testing (authentication, authorization, payment)
- User acceptance testing (UAT) with real users

Backup and Recovery Strategies:

- Daily database backups (automated)
- Weekly codebase backups to Git repository and local drives
- Server snapshot before deployment

- Disaster recovery plan: Restore from the latest backup in case of disk crash, application failure, or network issues

	Prepared By (Student)	Approved By (Teacher)
Date		

Design Plan:	Document Name:	SWD/Form No. 4
	GUI Standards Document	
Effective Date:	Version: 1	Page No.

Document Design

Property	Value
Document theme and color scheme	• Child-friendly, bright and fresh. • Main: #fff6e9, orange, brown.
Form - Background color	• #fff6e9 (main background)
Title - Font Size	• 1.5rem (large label) • 19px (sidebar menu) • 1.07rem - 1.5rem (book/event titles)
Title - Font Color	• #a66402 (highlight), #3d2e0e, #e28717 (orange shades)
Title - Font Style	• Bold, sans-serif, modern, friendly
Title – Alignment	• Center (main), left (sidebar/menu)
Background color of Controls on the form	• #fff4db (button normal) • #ff9800 (button highlight)
Foreground color of Controls on the form	• 1rem–16px (captions, labels)
Control Caption -- Font size	• #3d2e0e, #e28717, #ff9800
Control Caption - Font Color	• #3d2e0e, #e28717, #ff9800
Control Caption - Font Style	• Bold, easy to read, modern
Control caption and controls - Alignment	• Center (book, event, button), left (sidebar/menu)
Command button - Alignment	• Centered below book image, only visible on hover (for book cards)

	Prepared By (Student)	Approved By (Teacher)
Date		

Design Plan:	Document Name: Interface Design Document	SWD/Form No. 5
Effective Date:	Version: 1	Page No.

1. **List of forms to be created**

Document Name	Description	Controls on the Document
Homepage	Main landing page with book sections and events	Book grid, category menu, event carousel, search bar, AI chatbox
Login Form	User authentication	Input: email, password, Login button, create account button
Register Form	New user registration	Input: name, email, password, confirm password, Register button
Book List	Display all books by category, filter search	Book cards, filter dropdowns, pagination
Book Detail	Show details of selected book	Book info, add to cart, buy now, reviews, rating stars
Cart	Shopping cart view	List of books, quantity input, remove button, checkout
Checkout	Order confirmation and payment	Address input, payment method, place order button
Profile	User profile and personal info	Input: name, email, phone, edit/save button
Admin - User Management	Manage users (admin only)	User table, edit/delete buttons, search bar
Admin - Book Management	CRUD for books (admin only)	Book table, add/edit/delete buttons, search, pagination
Admin - Order Management	View/manage orders (admin only)	Order table, update status, view details button
Events/Promotions	List of bookstore events or promotions	Event cards, view details button
Contact/Support	Contact form for feedback/support	Input: name, email, message, send button

	Prepared By (Student)	Approved By (Teacher)
Date		

Table: users

Field Name	Data Type	Nul	Decistin
id	int(11)	No	User's name (uniqe)
name	varchar(100)	No	
emall	varchar(205)	No	User's password (hash)
password	varchar(255)	No	
role	enum(luser, admin)	No	User role Accourt creation time
created_at	timestamp	No	

Table: orders

Field Name	Data Type	Nul	Descrip
id	int(11)	No	Auto-inc oment order ID
user_id	int(11)	FK	
emall	varchar(255)	No	Foreign key to users(d)
name	varchar(255)	No	
phone	varchar(20)	No	Customer name
address	varchar(100)	No	
city	varchar(100)	No	Full addres
district	varchar(100)	No	

Table: books

Field Name	Data Type	Fay	Description
id	int(11)	No	Auto-increm-ent user ID
title	varchar(255)	No	Book title
author	varchar(255)	No	Book pnical
price	decimal(10.2)	No	Book price
description	text	No	Book description
image_url	varchar(255)	No	URL of book cover image
stock	int(11)	No	Quantity in stock
created_at	timestamp	No	Time order was
note	varchar(500)	Yes	Note for the order
payment_me mod	varchar(30)	No	var(saml(3y)
status	varchar(30)	No	Order status
created_at	daterime	No	Time order was

Table: orders

Field Name	Noil	Key
id	int(1)	PK
order id	int(1)	FK
book_id	int(1)	FK
quantity	int(1)	Per unit
price	decimol	Event tem tine

Table: events

Field Name	Noll	Key
id	int(1)	PK
order_id	int(11)	FK
title	int(11)	Title of
description	text	Event desciption
created_at	timestamp	Event creation

	Prepared By (Student)	Approved By (Teacher)
Date		

Design Plan:	Document Name: Process Design Document	SWD/Form No. 7
Effective Date:	Version: 1	Page No.

Flowchat
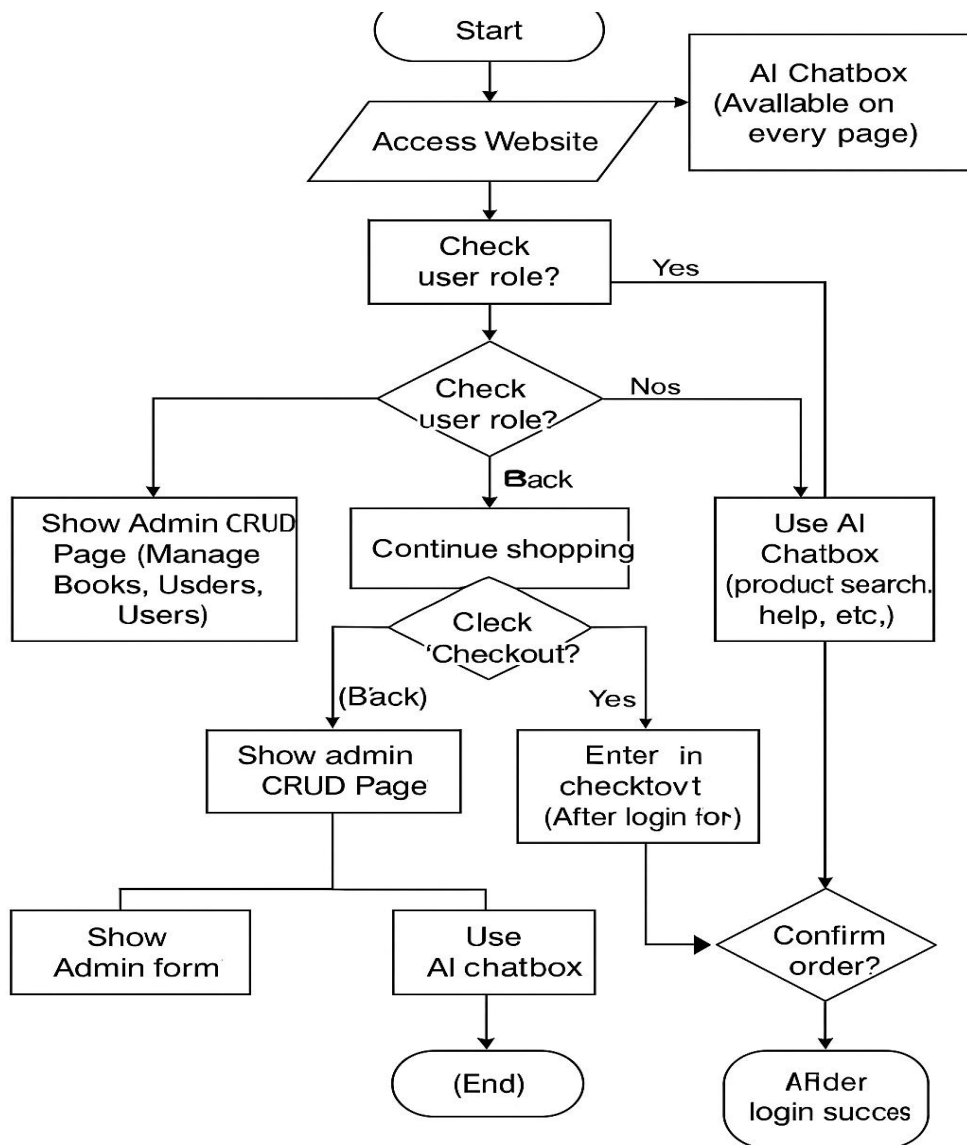
Date		
Design Plan:	**Document Name: Process Design Document**	**SWD/Form No. 7**
Effective Date:	**Version: 1**	**Page No.**

Entity Relationship (ER) Diagram

users
- id (PK)
- name
- email
- passwœrd
- role
- created_at

orders
- id (PK)   (PK)
- user_id (FK)
- email
- name
- phone
- address
- city
- district
- ward
- note
- payment_method
- status
- created_at

books
- id (PK)
- title
- author
- price
- description
- image_url
- stock
- created_at

order_items
- id (PK)   (PK)
- order_id (FK)   (FK)
- book_id (FK)   (FK)
- quantity
- price

events
- id (PK)
- title
- description
- created_at
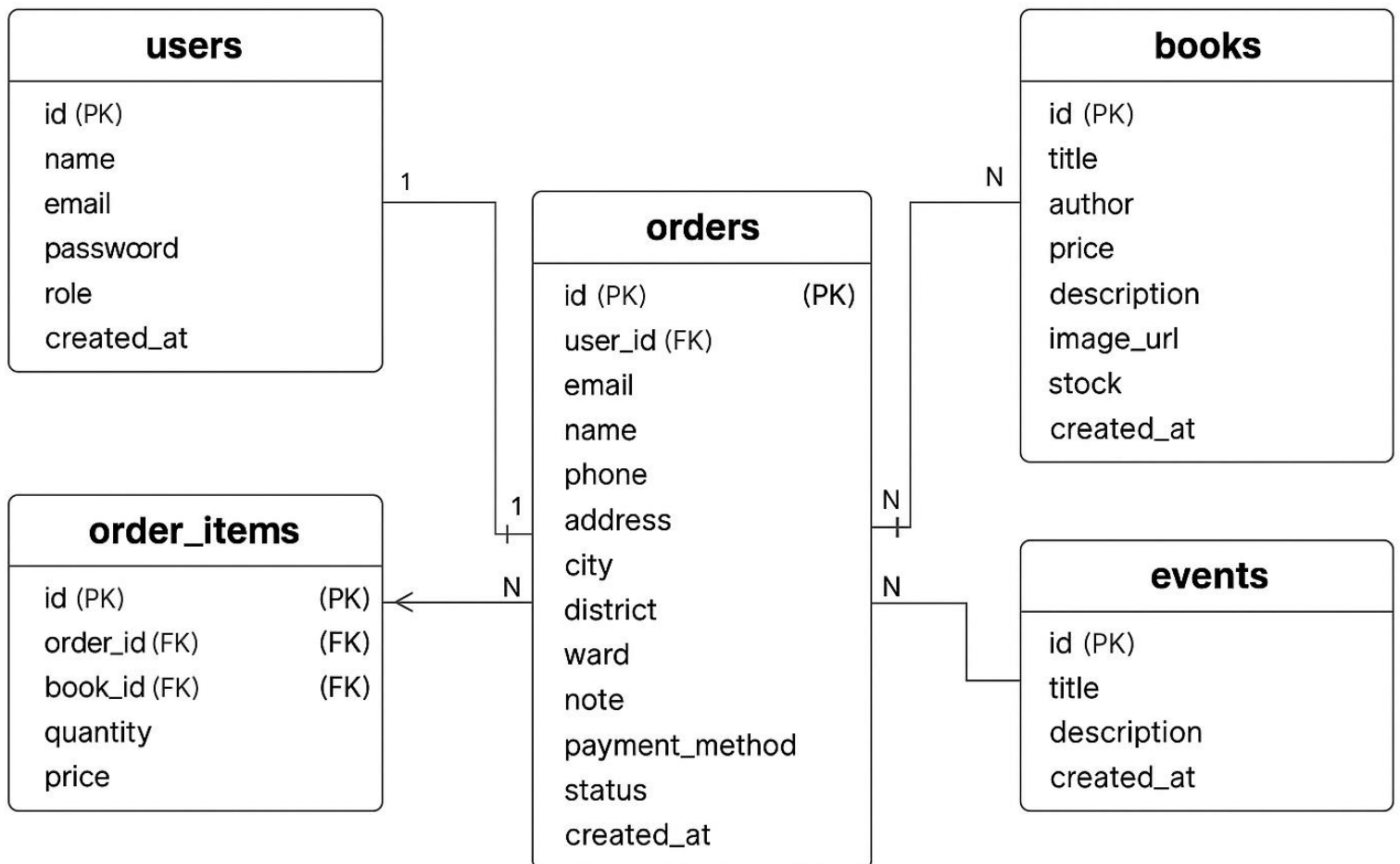
	Prepared By (Student)	**Approved By (Teacher)**
Date		

Architecture of the Application
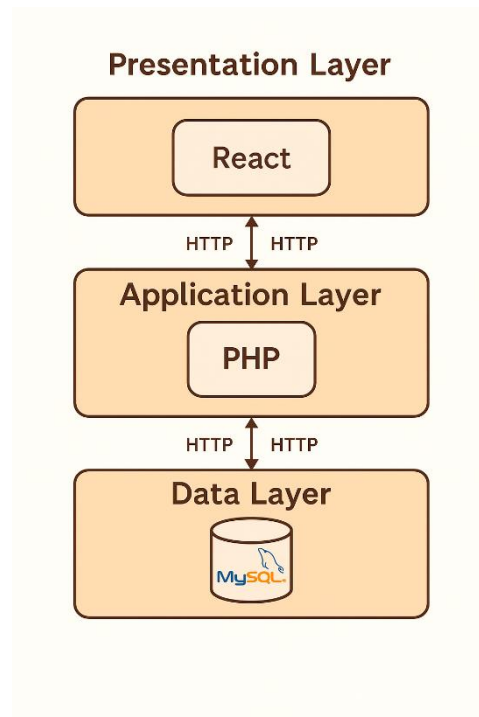
	Prepared By (Student)	Approved By (Teacher)
Date		

Data Flow Diagram
DFD LEVEL 0 - MediConnect

DFD Level 0

User request

System response

User

Bookiboo Web Application System
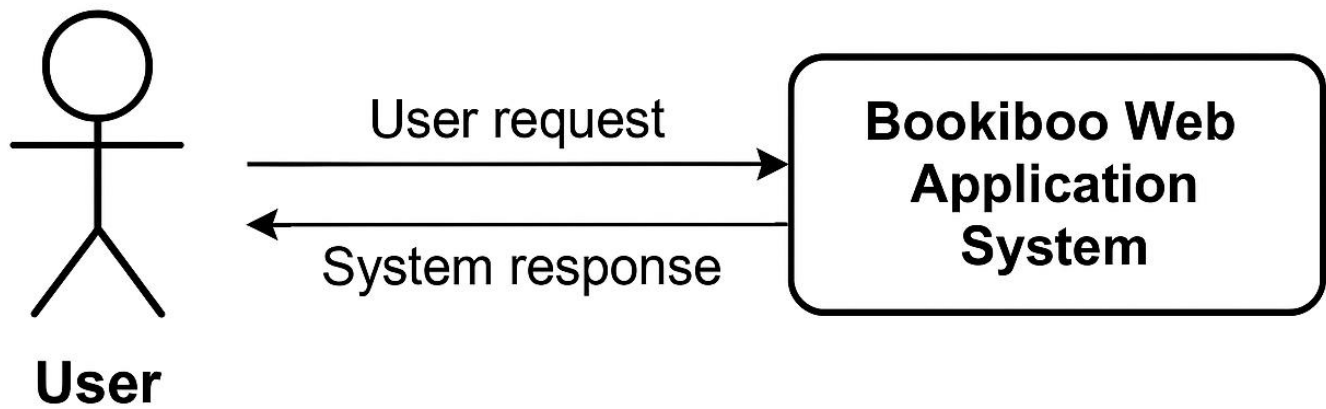
	Prepared By (Student)	Approved By (Teacher)
Date		

Data Flow Diagram
Figure 1.4: DFD LEVEL 1 - Main Processes

DFD Level 1
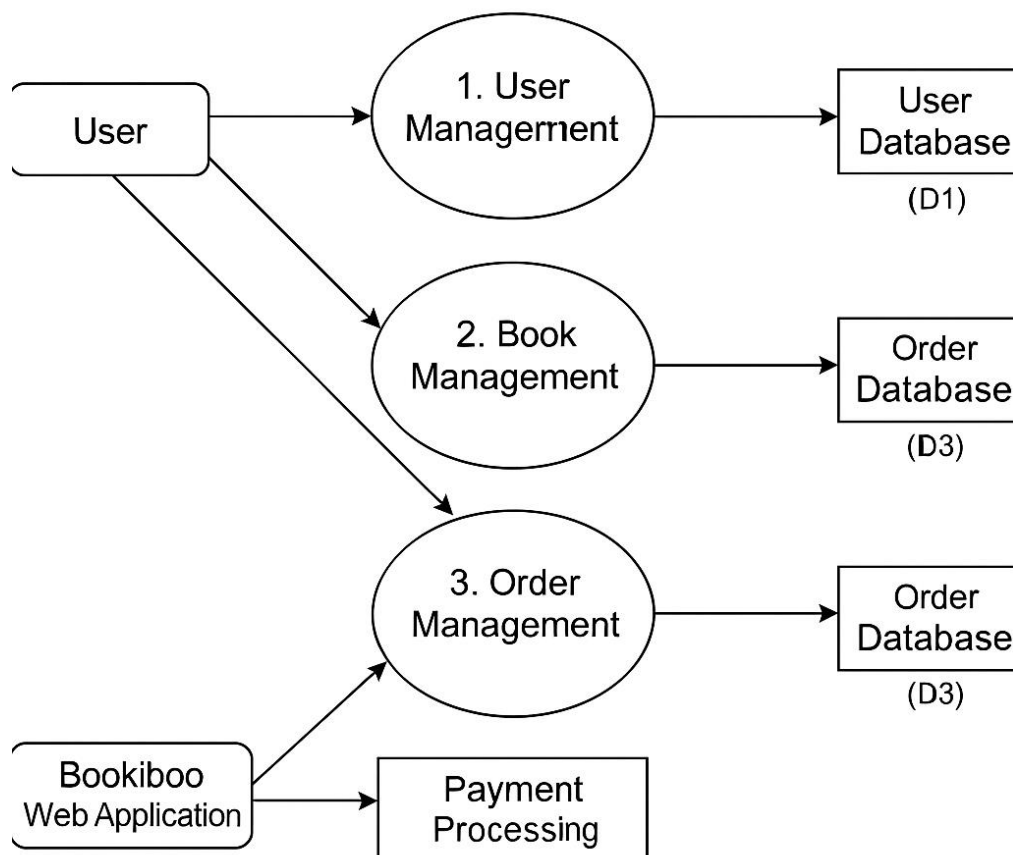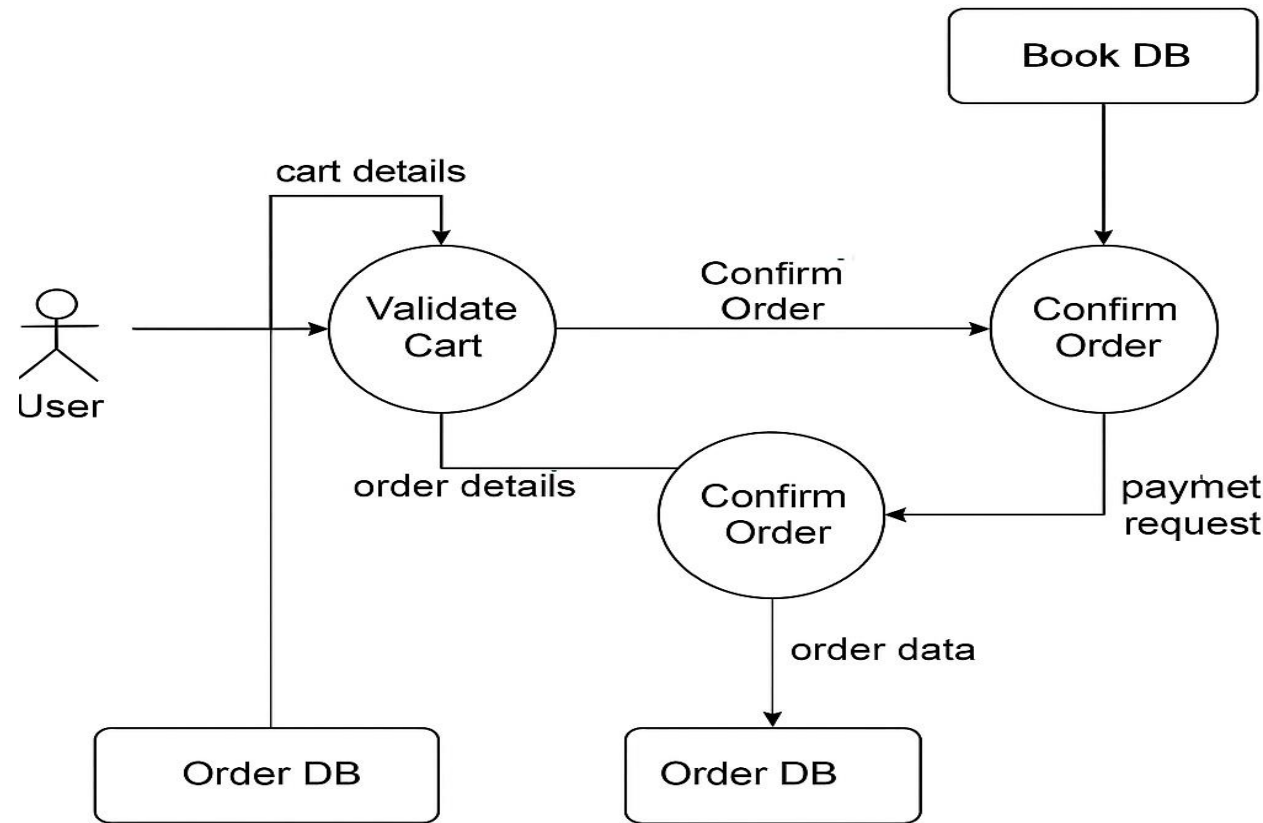
Data Flow Diagram

DFD Level 3

	Prepared By (Student)	Approved By (Teacher)
Date		
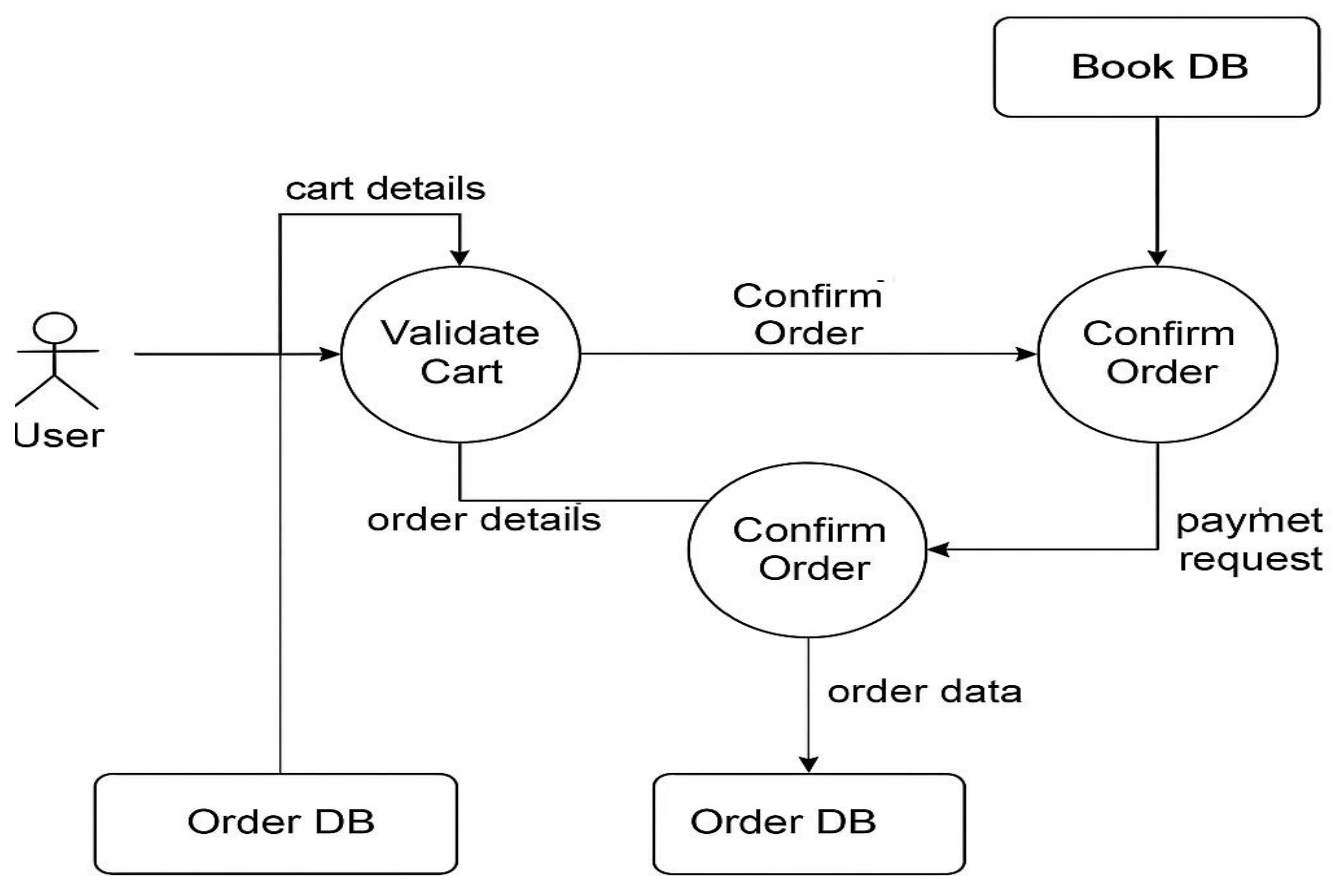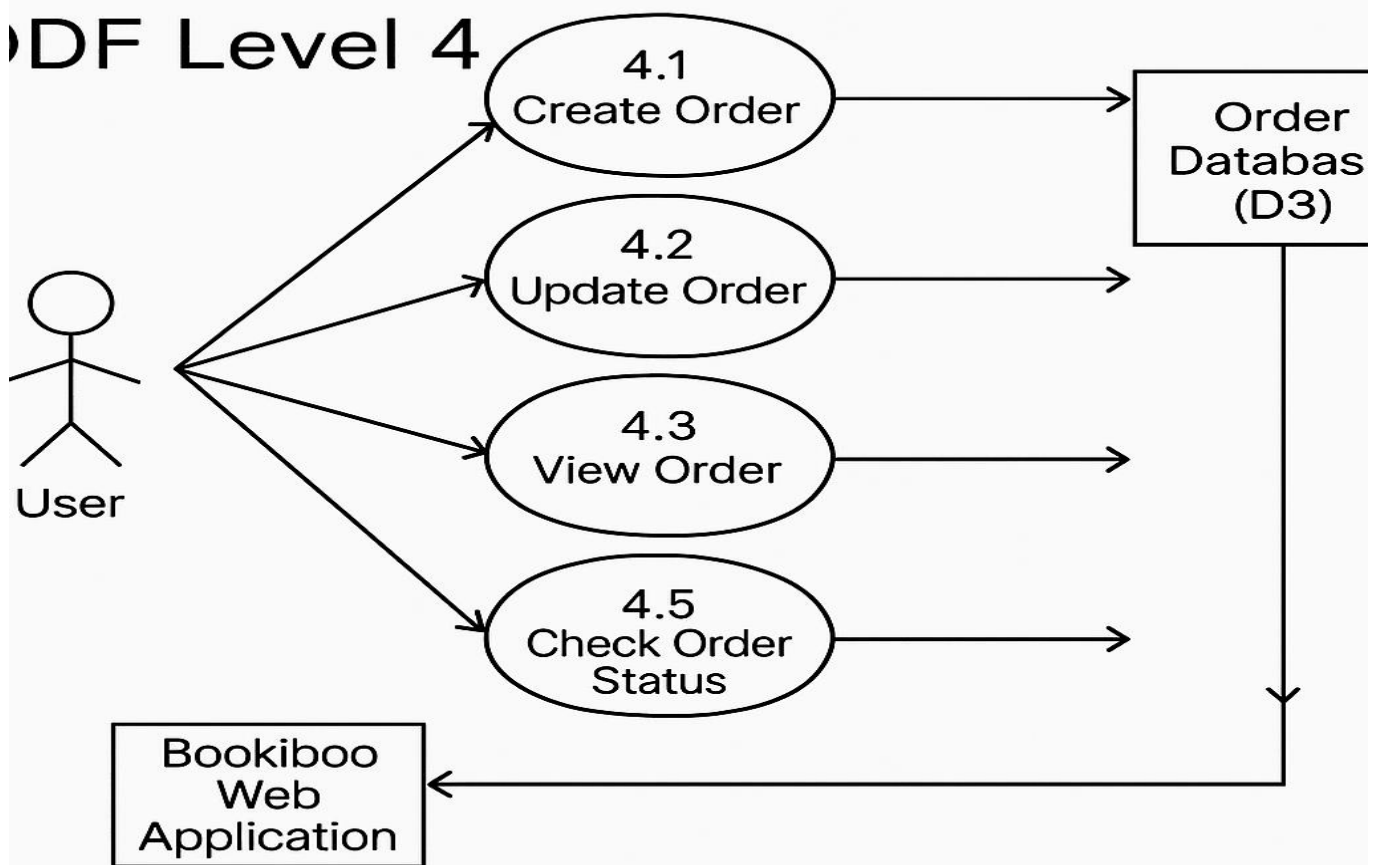
Data Flow Diagram

DFD Level 3

DF Level 4

User

4.1
Create Order

4.2
Update Order

4.3
View Order

4.5
Check Order
Status

Order
Databas
(D3)

Bookiboo
Web
Application

Design Plan:	Document Name: Process Design Document	SWD/Form No. 7
Effective Date:	Version: 1	Page No.

1. **Process Name**

        Bookibooo

2. **Process Description and Details**

This process defines the coding standards that all developers must follow when working on the Bookiboo web application project.

The purpose is to ensure code consistency, readability, maintainability, and to reduce errors during development.

These standards apply to all code written for the project, including frontend (ReactJS) and backend (PHP) components.

Regular code reviews will be conducted to ensure compliance with the standards described in this document.

	Prepared By (Student)	Approved By (Teacher)
Date		

Design Plan:	Document Name: Coding Standards Document	SWD/Form No. 8
Effective Date:	Version: 1	Page No.

1. Programming Standards

- All code must be clear, maintainable, and consistent with the project's architecture.
- Follow the MVC (Model-View-Controller) design pattern for backend and structured component-based design for frontend.
- Source code should be well-documented with comments explaining complex logic.
- Avoid code duplication by reusing functions and components where possible.

2. Standards for code writing style

- Use 2 or 4 spaces for indentation; do not use tabs.
- Each statement should be placed on a separate line.
- Opening braces { should be placed on the same line as the statement.
- Lines should not exceed 100 characters in length.
- Use blank lines to separate logical code blocks for readability.
- Always include a space after commas and around operators

3. Standards for declaring Variables

- Variable names must be meaningful and descriptive.
- Use **camelCase** for variable names (e.g., `totalPrice`, `userEmail`).
- Constant variables should be written in **UPPER_CASE** (e.g., `MAX_ITEMS`, `API_URL`).
- Do not use single-letter or ambiguous names, except for loop counters (`i`, `j`, `k`).
- Declare variables in the narrowest scope possible.
- Always initialize variables before using them.

4. Standards for Function Declarations

- Function names should be descriptive and use **camelCase** (e.g., `calculateTotal()`, `getUserInfo()`).
- Each function should perform a single, specific task.
- The function's purpose and parameters must be clearly commented, especially for complex logic.
- Limit the number of function parameters to 3–4 when possible; use objects for grouping related parameters.
- Functions should return values instead of modifying global variables whenever possible.
- Avoid deeply nested functions; keep functions as simple and short as possible.

5. Other Standards

- Write comments for all major logic blocks and important code sections.
- Every source file should begin with a header comment including file name, author, date, and description.
- Handle exceptions and errors properly, and avoid using empty catch blocks.

- Remove unused code, variables, and functions before committing.
- Use meaningful commit messages in version control.

- Follow the project's directory and file naming conventions.
- Regularly run code linting and formatting tools.

BOOKIBOO PROJECT – TASK ASSIGNMENT SHEET

No	Task Description	Assigned To	Start Date	End Date	Duration (days)	Status
1	Database Design	Loc	01-05-205	01-5-2025	5	Completed
2	Front-end Layout-^{*2} (PHP Core,AP|_Dmect)	Thu	02-05-205	08-5-2025	7	Completed
3	Admin Dashboard	Nguyen	03-05-205	10-5-2025	8	Completed
4	CRUD Books, Categeries, Orders	Loc	06-05-205	15-5-2025	8	Completed
5	CRUD Books. Catecnout Manament	Thu	19-05-205	19-5-2025	5	Completed
6	Event & News Management	Loc	20-05-205	21-5-2025	4	Completed
7	SEO, Security & BI Dashboted Optim.	Thu	18-05-205	24-5-2025	5	Completed
10	Final Testing & Pacaing, Report Prep.	Loc	19-05-205	21-5-2025	2	Completed
11	Seo, SETO, Security & BI Dashboard	Thu	20-05-205	24-5-2025	5	Completed
15	Final Sesting Packaging, Report Prep.	Nguyen	25-05-205	29-5-2025	3	In Progress
15	Final Testing & Packaging, Report Prep.	All Members	23-05-205	30-5-2025		

	Prepared By (Student)	Approved By (Teacher)
Date	Group 3	Lê Mộng Thúy

Test Document:	Document Name: Testing Document	SWD/Form No. 10
Effective Date:	Version: 1	Page No.

S.No.	Features Tested	Remarks
1	User Registration	Passed - Registration works fine
2	User Login	Passed - Login and logout OK
3	Browse Books	Passed - Book list displays
4	Add to Cart	Passed - Cart updates correctly
5	Checkout and Payment	Pending - Payment not integrated
6	Admin: Manage Books	Passed - Add/Edit/Delete works
7	Admin: Manage Orders	Passed - View & update status OK
8	Search and Filter Books	Passed - Search/filter works
9	Responsive Design	Passed - Looks good on mobile

	Prepared By (Student)	Approved By (Teacher)
Date		

Final Check List

(This document has to be filled by the teacher only. The teacher will do a complete functionality testing of the application and add his/her findings and suggestions to this form.)

Test Document:	Document Name: Final Check List	SWD/Form No. 12
Effective Date:	Version: 1	Page No.

S.No.	Aspect tested	Suggestions/ Remarks
1.	Have all the functionalities been properly integrated and are they completely functional?	
2.	Does each unit meet its objective and purpose? Are all the validations happening as specified in Process Design?	
3.	Have all Design and Coding standards been followed and implemented?	
4.	Is the GUI design consistent all over?	
5.	Are the codes working as per the specification?	
6.	Does the application's functionality resolve the client problem, and satisfy his/her requirements completely?	
7.	Have the hardware and software been correctly chosen?	
8.	Additional features and utilities that give value addition to the entire project.	

	Prepared By (Student)	Approved By (Teacher)
Date		

```
../bookiboo/Frontend/src/commo/LoginPage.js

import React, { useState } from "react";
import { useAuth } from "../hooks/useAuth";
import { useLocation, useNavigate } from "react-router-dom";


const LoginPage = ({ onClose }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const { login, auth } = useAuth();
  const navigate = useNavigate();
  const location = useLocation();

  const handleLogin = async (e) => {
    e.preventDefault();
    setError("");
    const result = await login(email, password);
    if (result && result.user?.role === "admin") {
      navigate("/admin/dashboard", { replace: true });
    } else if (result && result.user?.role === "user") {
      const redirectTo = location.state?.redirectTo || "/";
      navigate(redirectTo, { replace: true });
    } else {
      setError("email or pass is not correct!");
    }
  };



  const handleRegister = () => {
    navigate("/register");
  };

  return (
    <div style={{
      width: 400,
      margin: "60px auto",
      background: "#fff",
      borderRadius: 12,
      boxShadow: "0 2px 12px #ccc",
      padding: 32,
      position: "relative"
    }}>
      {onClose && (
        <button
          onClick={onClose}
          style={{
            position: "absolute", top: 20, right: 20,
```

```
            background: "transparent", border: "none", fontSize: 28, cursor:
"pointer"
          }}
          aria-label="Close"
        >
          ×
        </button>
      )}

      <h2 style={{ fontWeight: 700, marginBottom: 24 }}>Log In</h2>
      {error && (
        <div style={{ color: "red", marginBottom: 12, textAlign: "center"
}}>{error}</div>
      )}
      <form onSubmit={handleLogin}>
        <div style={{ marginBottom: 18 }}>
          <input
            type="email"
            placeholder="Email"
            value={email}
            autoComplete="username"
            onChange={e => setEmail(e.target.value)}
            required
            style={{
              width: "100%",
              padding: "18px 14px",
              borderRadius: 8,
              border: "none",
              background: "#eee",
              fontSize: 17,
              marginBottom: 2,
              outline: "none"
            }}
          />
        </div>
        <div style={{ marginBottom: 18 }}>
          <input
            type="password"
            placeholder="Password"
            value={password}
            autoComplete="current-password"
            onChange={e => setPassword(e.target.value)}
            required
            style={{
              width: "100%",
              padding: "18px 14px",
              borderRadius: 8,
              border: "none",
              background: "#eee",
              fontSize: 17,
              marginBottom: 2,
              outline: "none"
            }}
```

```
          />
        </div>
        <button
          type="submit"
          style={{
            width: "100%",
            padding: "16px 0",
            border: "none",
            borderRadius: 6,
            background: "#e12727",
            color: "#fff",
            fontSize: 20,
            fontWeight: "bold",
            marginBottom: 14,
            cursor: "pointer",
            transition: "background 0.2s"
          }}
        >
          Login
        </button>
      </form>
      <button
        onClick={handleRegister}
        style={{
          width: "100%",
          padding: "14px 0",
          border: "2px solid #8d8d8d",
          borderRadius: 6,
          background: "#fff",
          color: "#111",
          fontSize: 20,
          fontWeight: "bold",
          cursor: "pointer",
          transition: "background 0.2s"
        }}
      >
        Create An Account
      </button>
    </div>
  );
};

export default Logi
```