* Database Requirements:
* Book Information:
  + Title
  + Author(s)
  + Genre
  + ISBN (International Standard Book Number)
  + Publication Year
  + Description
  + Price
  + Quantity Available
  + Format (e-book, hardcopy, etc.)
  + Availability Status
* User Information:
  + Student ID (for students)
  + Staff ID (for staff)
  + Name
  + Contact Information
  + Email
  + Address
* Transaction History:
  + Transaction ID
  + User ID
  + Book(s) purchased (linked to Book Information)
  + Date and Time of Transaction
  + Total Amount
  + Transaction Status (completed, pending, canceled)
* Inventory Management:
  + Tracking of available stock for each book
  + Notification system for low stock
  + Reorder point for automatic restocking
* Security:
  + User authentication and authorization
  + Encrypted storage of sensitive user information
  + Secure transaction handling
* Search and Filtering:
  + Ability to search books by title, author, genre, etc.
  + Filtering options for users to refine search results
* Shopping Cart:
  + Temporary storage of selected items before checkout
  + Modification and removal of items from the cart
* Order Processing:
  + Confirmation emails to users after successful transactions
  + Order fulfillment process to update inventory and complete transactions
* Reporting and Analytics:
  + Generate reports on popular books, sales trends, etc.
  + Analytical tools for strategic decision-making
* Responsive User Interface:
  + Intuitive and user-friendly interface for both students and staff
  + Mobile-friendly design for accessibility
* Implementation Steps:
* Data Modeling:
  + Identify entities, relationships, and attributes
  + Create an Entity-Relationship Diagram (ERD)
* Normalization:
  + Apply normalization techniques to minimize redundancy and improve efficiency
* Database Design:
  + Choose an appropriate database management system (e.g., My, Postgre)
  + Define tables, relationships, and constraints based on the ERD
* Front-end Development:
  + Design and implement a user interface for the bookstore
  + Ensure compatibility with various devices and browsers
* Back-end Development:
  + Develop server-side scripts and APIs to handle user requests and database interactions
* Testing:
  + Conduct thorough testing of the system, including unit tests, integration tests, and user acceptance tests
* Deployment:
  + Deploy the database system to a secure server
  + Monitor system performance and address any issues
* Training and Documentation:
  + Provide training for bookstore staff on system usage
  + Create comprehensive documentation for future reference
* Enhancements to the Database System:
* Publisher Information:
  + Capture details of publishers sending lists of latest books and materials.
  + Track orders placed with publishers, including order date and delivery status.
* Warehouse Management:
  + Integrate a module for warehouse management to record the arrival of books.
  + Associate received books with corresponding orders.
* Website Update:
  + Implement a mechanism to automatically update the website with new book information once the clerk records the arrival in the warehouse.
* Invoice Generation:
  + Develop an invoice generation system triggered by the arrival of new books.
  + Link generated invoices to corresponding transactions.
* Accounts Integration:
  + Establish a seamless connection with the accounts department for invoice processing and payment.
* Customer Registration:
  + Include a registration module for customers to become members.
  + Capture member details like name, contact information, and delivery address.
* Review and Comparison:
  + Enable members to read and write reviews for books.
  + Implement a feature for customers to compare products.
* Shopping Cart Functionality:
  + Design and implement a shopping cart system for members to add books for purchase.
  + Provide a summary of the selection and the total cost before checkout.
* Payment Gateway:
  + Integrate a secure payment gateway for online transactions.
  + Ensure the system supports various payment methods.
* Order Processing:
  + Develop an order processing system to handle customer requests.
  + Send order confirmation emails to customers.
* Receipt Generation:
  + Generate and allow customers to print or save receipts upon successful payment.
* Delivery Tracking:
  + Implement a system to track the status of book deliveries.
  + Notify customers about the delivery status and expected delivery date.
* Feedback and Ratings:
  + Create a module for users to provide feedback on books, including scores and optional short text.
  + Ensure restrictions on changes to feedback and limit to one per user per book.
* Data Analytics:
  + Implement tools for analyzing user opinions, book ratings, and feedback.
  + Use analytics to identify popular books and improve the bookstore's offerings.
* Security and Privacy:
  + Implement robust security measures to protect customer data and financial transactions.
  + Ensure compliance with privacy regulations.
    - Design and Implementation:
* Entity-Relationship (ER) Diagram:
  + Entities:
  + Book (BookID [PK], Title, Author, Genre, ISBN, PublicationYear, Description, Price, Format, AvailabilityStatus)
  + Publisher (PublisherID [PK], Name, Address, Contact)
  + Order (OrderID [PK], OrderDate, DeliveryStatus)
  + Customer (CustomerID [PK], Name, Address, Contact, MemberID)
  + Invoice (InvoiceID [PK], InvoiceDate, TotalAmount)
  + Feedback (FeedbackID [PK], Score, Comment)
  + Transaction (TransactionID [PK], TransactionDate, TotalAmount)
  + Relationships:
  + Book-Publisher (Many-to-One): Many books can be published by one publisher.
  + Order-Book (One-to-Many): One order can consist of multiple books.
  + Order-Publisher (Many-to-One): Many orders can be placed with one publisher.
  + Customer-Transaction (One-to-Many): One customer can make multiple transactions.
  + Customer-Invoice (One-to-Many): One customer can have multiple invoices.
  + Customer-Feedback (One-to-Many): One customer can provide multiple feedback.
  + Transaction-Invoice (One-to-One): Each transaction corresponds to one invoice.
  + Book-Feedback (One-to-Many): One book can have multiple feedback.
  + Business Rules:
  + Each book must be associated with at least one publisher.
  + An order must consist of at least one book.
  + Each order is placed with one specific publisher.
  + Each transaction corresponds to one invoice.
  + Each customer can provide multiple feedback for different books.
* Relational Schema and Normalization:
  + Book (BookID [PK], Title, Author, Genre, ISBN, PublicationYear, Description, Price, Format, AvailabilityStatus, PublisherID [FK])
  + Publisher (PublisherID [PK], Name, Address, Contact)
  + Order (OrderID [PK], OrderDate, DeliveryStatus, PublisherID [FK])
  + Customer (CustomerID [PK], Name, Address, Contact, MemberID)
  + Invoice (InvoiceID [PK], InvoiceDate, TotalAmount, CustomerID [FK])
  + Feedback (FeedbackID [PK], Score, Comment, CustomerID [FK], BookID [FK])
  + Transaction (TransactionID [PK], TransactionDate, TotalAmount, InvoiceID [FK])
* Data Dictionary:
  + [Refer to the relational schema above for attributes, data types, and constraints]
* Normalization Process:
  + 1NF: All attributes are atomic.
  + 2NF: No partial dependencies on the primary key.
  + 3NF: No transitive dependencies on the primary key.
* DDL Implementation:
  + Create Tables CREATE TABLE Book ( BookID INT PRIMARY KEY, Title VARCHAR(255), Author VARCHAR(255), -- ... (Other attributes) PublisherID INT REFERENCES Publisher(PublisherID) ); CREATE TABLE Publisher ( PublisherID INT PRIMARY KEY, Name VARCHAR(255), Address VARCHAR(255), Contact VARCHAR(20) ); -- Similar CREATE TABLE statements for Order, Customer, Invoice, Feedback, Transaction -- Enforce Integrity (Primary and Foreign Keys) ALTER TABLE Book ADD FOREIGN KEY (PublisherID) REFERENCES Publisher(PublisherID); -- Similar ALTER TABLE statements for other Foreign Keys.
* b) Data Manipulation Language (DML) Queries:
  + - Latest Books by Various Publishers:
* SELECT Publisher.Name, Book.Title, MAX(Book.PublicationYear) AS LatestPublicationYear FROM Publisher JOIN Book ON Publisher.PublisherID = Book.PublisherID GROUP BY Publisher.Name, Book.Title;
  + - Books Ordered by Store Manager:
* SELECT Order.OrderDate, Book.Title, Order.DeliveryStatus FROM Order JOIN Book ON Order.OrderID = Book.OrderID;
  + - Invoices for Various Publishers:
* SELECT Invoice.InvoiceNumber, Invoice.InvoiceDate, Publisher.Name, Publisher.Address, SUM(Book.Price) AS TotalAmount FROM Invoice JOIN Transaction ON Invoice.TransactionID = Transaction.TransactionID JOIN Order ON Transaction.OrderID = Order.OrderID JOIN Publisher ON Order.PublisherID = Publisher.PublisherID JOIN Book ON Order.OrderID = Book.OrderID GROUP BY Invoice.InvoiceNumber, Invoice.InvoiceDate, Publisher.Name, Publisher.Address;
  + - List of Registered Customers:
* SELECT Customer.MemberID, Customer.Name, Customer.Address, Customer.Contact FROM Customer;
  + - Purchased Books Delivered to Members:
* SELECT Customer.MemberID, Customer.Name, Customer.Address, Customer.Contact, Book.Title, Transaction.TransactionDate, Transaction.DeliveryStatus FROM Customer JOIN Transaction ON Customer.CustomerID = Transaction.CustomerID JOIN Invoice ON Transaction.InvoiceID = Invoice.InvoiceID JOIN Book ON Transaction.BookID = Book.BookID;
  + - List of Books by Specific Categories:
* SELECT Genre, Book.Title, Book.Quantity FROM Book WHERE Genre = 'Science Fiction';
  + - Total Number of Books for Each Category:
* SELECT Genre, COUNT(\*) AS TotalBooks FROM Book GROUP BY Genre;
  + - Total Books and Total Price in Shopping Cart:
* SELECT COUNT(\*) AS TotalBooks, SUM(Book.Price) AS TotalPrice FROM Book WHERE Book.InShoppingCart = true;
  + - List of Books with Feedback Count:
* SELECT Book.Title, COUNT(Feedback.FeedbackID) AS FeedbackCount FROM Book LEFT JOIN Feedback ON Book.BookID = Feedback.BookID GROUP BY Book.Title;

1. Database and Database Management System
2. Disadvantages of File-Based System:
3. Data Redundancy: In a file-based system, data is often duplicated across various files, leading to redundancy and increased storage requirements.
4. Data Inconsistency: Changes made to data in one file may not be reflected in other files, resulting in inconsistencies.
5. Limited Data Sharing: File-based systems make it challenging to share data among different departments or users, limiting collaboration.
6. Data Dependence: Applications are highly dependent on the structure of data files, making it difficult to modify or update without affecting existing programs.
7. Data Integrity: Ensuring data integrity is complex, as there is no centralized control over data validation and enforcement.
8. Advantages of Database and DBMS:
9. Data Centralization: Databases centralize data storage, reducing redundancy and promoting data consistency.
10. Data Independence: Database systems provide a layer of abstraction, allowing changes to the database structure without affecting applications.
11. Efficient Data Retrieval: DBMS provides powerful query languages for efficient data retrieval based on various criteria.
12. Data Security: Database systems implement access controls to restrict unauthorized access and ensure data security.
13. Concurrency Control: DBMS manages concurrent access to data, preventing conflicts and ensuring data consistency.
14. Data Integrity and Validation: Database systems enforce data integrity constraints and support validation mechanisms.
15. Data Sharing: Multiple users can access and share data simultaneously, facilitating collaboration.
16. Scalability: Database systems are scalable, accommodating growth in data volume and user access.
17. Relating to the Case Study:
18. In the case study, the TU E-Bookstore faces challenges with limited availability and restrictions in the current file-based system. Adopting a database and DBMS will address these issues by centralizing data, improving data consistency, and enabling efficient data retrieval. It will also support scalability as the bookstore expands.
19. Business Rules & Normalization
20. Business Rules:
21. Each book must be associated with at least one publisher.
22. An order must consist of at least one book.
23. Each order is placed with one specific publisher.
24. Each transaction corresponds to one invoice.
25. Each customer can provide multiple feedback for different books.
26. Example of UNF (Unnormalized Form):
27. Consider a table containing customer and book information without normalization:
28. UNF Table: Order\_Customer\_Book
29. OrderID, CustomerID, CustomerName, BookID, BookTitle, OrderDate, Quantity, Price
30. Normalization Steps (up to 3NF):
31. 1NF:
32. Break the table into separate tables for Orders, Customers, and Books.
33. 2NF:
34. Identify and remove partial dependencies.
35. Create tables for OrderDetails and BookDetails.
36. 3NF:
37. Identify and remove transitive dependencies.
38. Create tables for Customers and Books.
39. Entity Relationship Diagram (ERD)
40. Design the Database Using Chen’s or Crow’s Foot Notation:
41. [Provide the ERD using Crow’s Foot Notation]
42. Database Schema
43. Finalized ERD Using Crow’s Foot Notation:
44. [Provide the finalized ERD]
45. Generate the Database Diagram from the DBMS:
46. [Include the database diagram generated from the DBMS]
47. SQL-Data Definition Language (DDL)
48. Create Tables with Suitable Data Types:
49. [Include DDL statements for creating tables]
50. Insert 5-10 Rows of Data into Each Table:
51. [Include SQL statements for data insertion]
52. Screen Shot All Tables with Data:
53. [Include screenshots of all tables with data]
54. Screen Shot All Query Statements:
55. [Include screenshots of SQL query statements]
56. SQL-Data Manipulation Language (DML)
57. Write SQL Statements to Answer Questions: i. [Include SQL statement for the list of latest books sent by various publishers based on a monthly basis.] ii. [Include SQL statement for a list of books ordered by the store manager from various publishers based on a monthly basis.] iii. [Include SQL statement for producing a record of invoices for various publishers.]