**Phase 1: Project Setup and Initial Planning**

**Focus:** Initial planning and setting up the environment.

**Sub-Points:**

* **Project Scope & Requirements:** Define clear goals (e.g., manage books, users, transactions).
* **Database Design:** Design a scalable relational database schema (e.g., MySQL, PostgreSQL) for books, users, transactions, fines, etc.
  + Define relationships (One-to-many, many-to-many) for books, categories, and users.
  + Plan for scalability (future-proofing with new book categories, member tiers).
* **User Roles:** Define user roles and permissions (e.g., Admin, Librarian, Member).
  + Determine what each role can and cannot access (CRUD operations on books, member management, etc.).
* **Tech Stack Selection:** Choose technologies for backend (Node.js, Python/Django, etc.), frontend (React, Angular, etc.), and database (SQL or NoSQL).
* **Version Control Setup:** Set up version control using Git and a repository on GitHub/GitLab.
* **Initial Wireframes/Prototypes:** Sketch basic UI/UX flow for the application (login screen, dashboard).

**Phase 2: User Authentication System (Login/Signup)**

**Focus:** Login system with different user roles.

**Sub-Points:**

* **User Registration:** Implement sign-up functionality for new users.
  + Basic details like name, email, password.
  + Validate email format, password strength.
* **Login System:** Implement secure login functionality.
  + Use session-based (cookies) or JWT (for stateless authentication) for session management.
* **User Roles & Permissions:**
  + Different roles (Admin, Librarian, Member) with different access levels.
  + Admin can add/remove books, manage users, and view reports.
  + Librarian can manage book transactions (borrow/return).
  + Member can view books, borrow/return, check fines, etc.
* **Forgot Password/Reset:** Implement "Forgot Password" and secure password reset.
* **Multi-Factor Authentication (MFA):** Optionally implement for added security, especially for admins.
* **Security Considerations:**
  + Use hashing (bcrypt) for storing passwords.
  + Secure API endpoints with role-based access control (RBAC).

**Phase 3: Adding Books and Managing Members**

**Focus:** Basic CRUD operations for books and member registration.

**Sub-Points:**

* **Book CRUD (Create, Read, Update, Delete):**
  + Allow librarians to add new books to the system (title, author, category, ISBN, etc.).
  + Implement searching/filtering for books by category, author, etc.
* **Member Registration:** Admin or Librarians can register new members.
  + Collect member details (name, contact info, membership type, etc.).
* **Book Categories:** Define book categories (e.g., Fiction, Science, History, etc.).
* **Book Stock Management:**
  + Track the number of available copies for each book.
  + Update stock upon borrowing/returning.
* **Member Management:**
  + Admins can view/update member profiles.
  + Track membership type (e.g., student, regular).
  + Manage member activity (e.g., overdue fines, borrowing limits).

**Phase 4: Book Borrowing and Transactions**

**Focus:** Manage borrowing/returning books and transactions.

**Sub-Points:**

* **Borrowing Books:**
  + Implement book borrowing functionality with date tracking (borrow date, due date).
  + Enforce borrowing limits based on user role/type.
* **Returning Books:**
  + Track book returns and update availability.
  + Validate if the returned book was actually borrowed.
* **Transaction History:**
  + Record all transactions (borrow/return) with timestamps and details.
  + Show transaction history for members (borrowed books, due dates, fines).
* **Book Availability Status:**
  + Show real-time status of book availability (Available, Borrowed, Reserved).
  + Prevent borrowing of books that are not available.

**Phase 5: Fine Calculation for Overdue Books**

**Focus:** Implement fine calculations and tracking for overdue books.

**Sub-Points:**

* **Fine Calculation Logic:**
  + Implement logic to calculate overdue fines (e.g., daily late fee, flat fine per day).
  + Fine rates can be dynamic (e.g., different rates for different book categories or membership types).
* **Integration with Transactions:**
  + Track overdue fines as part of the transaction history.
  + Display overdue fines on member profile/dashboard.
* **Payment System (Optional for Phase 5):**
  + Allow users to pay fines online or in-person.
  + Record payments and update fine status.
* **Notifications:**
  + Send automated notifications to users when fines are due or overdue.

**Phase 6: Book Reservations and Renewals**

**Focus:** Reservations for borrowed books and renewal functionality.

**Sub-Points:**

* **Book Reservations:**
  + Allow members to reserve books that are currently borrowed by others.
  + Notify users when the reserved book is available.
* **Book Renewals:**
  + Allow users to renew borrowed books if no one has reserved it.
  + Track renewal date and avoid multiple renewals on the same book by the same member.
* **Availability Management:**
  + Ensure reserved books are automatically marked as unavailable for others.
  + Manage renewals and reservations in the system without causing conflicts.

**Phase 7: Reporting and Admin Dashboard**

**Focus:** Create reports for admins and managers.

**Sub-Points:**

* **Admin Dashboard:**
  + Display key metrics like the number of books, active members, overdue books, fines, etc.
  + Display transactional data (e.g., borrowed/returned books).
* **Report Generation:**
  + Allow filtering reports by date ranges, book categories, user types.
  + Provide downloadable reports (CSV/PDF).
* **Visualizations:**
  + Integrate charts/graphs for insights (e.g., number of books borrowed per month, overdue fines).
  + Visualize data for decision-making (e.g., most popular books).

**Phase 8: Final Testing, Security, and Deployment**

**Focus:** Final testing, security improvements, and deployment.

**Sub-Points:**

* **Testing:**
  + Unit Testing: Write tests for all major functionalities (book borrowing, fine calculations, etc.).
  + Integration Testing: Ensure modules work together (user authentication + transactions).
  + End-to-End Testing: Simulate user workflows to test the system.
  + Performance Testing: Ensure system handles a high number of concurrent users.
* **Security Improvements:**
  + Secure API endpoints and ensure data encryption (SSL/TLS).
  + Ensure proper input sanitization to prevent SQL injection or XSS attacks.
* **Deployment:**
  + Choose a hosting provider (AWS, Azure, DigitalOcean, etc.).
  + Set up CI/CD pipelines for automatic deployment.
  + Monitor system performance and ensure backups are in place.