Here’s the complete “map” of everything we built, how the PHP files relate to one another, and a bird’s-eye view of the system flow—from your very first book-exchange page through to in-app notifications, requests, accepts, rejects, and chat:

**1. Authentication**

* **user\_login.php**
  + Displays login form; on POST checks credentials (plaintext or hashed), sets $\_SESSION['user\_id'] & $\_SESSION['first\_name'], then redirects to booksexchange.php.
* **logout.php**
  + Clears the session and redirects to login or homepage.

**2. Main Book-Exchange Interface**

* **booksexchange.php**
  + **Fetches** from book\_exchange + department names + a LEFT JOIN subquery counting rows in book\_offers to show how many copies are available.
  + **Renders** a grid of “book cards,” each with “Give” and “Take” buttons.
  + **Injects** two Bootstrap modals:
    - **Give Modal** (form → POST to give.php)
    - **Take Modal** (populated dynamically via AJAX from getOffers.php)
  + **JS logic** for: category filters, search, wiring the modals, sending AJAX to take.php, drop.php, etc.
  + Uses query‐string flags (e.g. ?offer=exists or ?offer=ok) plus a tiny inline script to show one-time alerts upon page load.

**3. Offering Books**

* **give.php**
  + **Checks** “must be logged in.”
  + **Prevents duplicates** by querying book\_offers for an existing (user\_id, book\_id, desired\_book\_id IS NULL).
  + If duplicate: redirect back to booksexchange.php?offer=exists.
  + Otherwise INSERT into book\_offers(book\_id, user\_id, details), then redirect back to booksexchange.php?offer=ok.

**4. Viewing & Dropping Offers**

* **getOffers.php**
  + AJAX endpoint: takes ?book\_id=…, SELECTs all offers on that book, JOINing users to get giver\_name & giver\_id, plus details and offered\_at.
  + Returns JSON array of objects: { offer\_id, giver\_id, giver\_name, book\_condition, offered\_at }.
* **drop.php**
  + AJAX POST: takes offer\_id, verifies user\_id owns that offer, deletes it from book\_offers, re-counts remaining offers for that book, and returns JSON { status:'ok', book\_id, copies }.

**5. Requesting Books**

* **take.php**
  + AJAX POST: takes offer\_id, verifies logged in, checks book\_requests for an existing (offer\_id, requester\_id) to prevent duplicates—if exists, returns {"status":"exists"}.
  + Otherwise INSERTs into book\_requests(offer\_id, requester\_id, status='pending').
  + **Creates** a row in notifications for the *owner* of that offer:

sql

CopyEdit

INSERT INTO notifications

(user\_id, receiver\_id, action\_id, type, message)

VALUES

(requester\_id, owner\_id, request\_id, 'book\_request', '… has requested your book');

* + Returns JSON {"status":"ok"}.

**6. Notifications Infrastructure**

* **Database table** notifications holds:
  + id, user\_id, receiver\_id, action\_id, type, message, is\_read, created\_at
* **Header partial** (includes/header.php)
  + At session start, if $\_SESSION['user\_id'] is set, does:

php

CopyEdit

SELECT COUNT(\*) AS cnt

FROM notifications

WHERE receiver\_id = ?

AND is\_read = 0

* + Renders a bell-icon link with a little red badge showing that unread count.
  + Tapping it toggles an **offcanvas panel** (see below).
* **Offcanvas panel snippet** (inserted just after <body> or in your layout):
  + Loads an array $panelNotes (same query as header, but fetches full rows):

php

CopyEdit

SELECT \*

FROM notifications

WHERE receiver\_id = ?

ORDER BY created\_at DESC

* + Loops over them, marking read/unread visually, and for those of type book\_request shows **Accept** & **Reject** buttons.
  + JS wires up:
    - **Accept →** window.location.href = 'chat.php?request\_id=…';
    - **Reject →** AJAX POST → reject\_request.php, then removes that list item on success.
* **reject\_request.php**
  + AJAX endpoint: takes request\_id, marks the matching book\_requests row as status='rejected' (or deletes it), returns JSON {"status":"ok"}.

**7. Full Notifications Page**

* **notifications.php**
  + Marks all unread notifications for receiver\_id as is\_read=1.
  + SELECTs them all, newest first, and lists them with timestamps and messages.

**8. Chat Handoff**

* **chat.php?request\_id=…**
  + (Your next step) should load the two users involved—via the book\_requests → book\_offers → users chain—and present a messaging UI.

**How It All Fits Together**

1. **User logs in** (user\_login.php → session).
2. **User lands on** booksexchange.php, sees books with availability, can click **Give** or **Take**.
3. **Give:** modal → give.php → inserts your offer → you get a success alert back on booksexchange.php.
4. **Take:** modal opens → AJAX getOffers.php shows existing offers; click **Request** → AJAX take.php → inserts book\_requests, inserts a notifications row for the offer-owner.
5. **Offer-owner** sees a red badge on the bell in the header; clicks it → offcanvas appears showing “X requested your book.”
6. They click **Accept** (→ chat.php) or **Reject** (→ reject\_request.php AJAX), which updates the DB, removes the panel item, etc.
7. From the bell you can also click “See all” → full notifications.php page.

**1. Database & Schema**

* **chats** table linking two users (user1\_id, user2\_id) and the original request\_id.
* **chat\_messages** table storing (chat\_id, sender\_id, receiver\_id, message, timestamp).
* **notifications** table capturing both “request\_accepted” and “new\_message” events.

**2. Request → Chat Flow**

* **accept\_request.php**: owner clicks “Accept,” we mark the request accepted, insert a chats row, send a “request\_accepted” notification, and return the new chat\_id.
* **reject\_request.php**: owner clicks “Reject,” we mark that request as rejected and remove it from the panel.

**3. Chat UI (in chat.php)**

* A modern Bootstrap off-canvas–style chat panel: header, scrollable message area, sticky footer with input.
* **send\_message.php** inserts a new chat\_messages row and creates a “new\_message” notification for the receiver.
* **get\_messages.php** (AJAX endpoint) returns JSON of messages for a chat (optionally since a given timestamp), letting the front-end poll every 3 s to append only new messages without a full reload.

**4. Notifications Panel (offcanvas)**

* Lists up to 5 most recent notifications (both request and message types).
* **Accept/Reject** buttons for pending book-requests.
* **Click-to-chat** on message notifications.
* **Per-notification “×”** to soft-delete a single item (AJAX → delete\_notification.php).
* **“Clear All”** button to soft-delete every notification (AJAX → clear\_notifications.php).

**5. Recent Chats List**

* A standalone **recent\_chats.php** page showing each conversation partner exactly once, with:
  + Their name & avatar
  + The last message snippet
  + Timestamp of that last message
  + A badge count of unread messages
* Front-end JS fetches **chats\_list.php** (JSON) and populates the list on page load or when you click the refresh icon.
* Clicking any entry opens chat.php?chat\_id=….