1 Introduction and Use Cases

Your employer, Hypothetical Books (HB), is a large independent bookstore. Thanks to a loyal customer following in the community, they’ve grown large enough that the mishmash of spreadsheets used to track their inventory and organize their store is no longer sufficient. Your group has been contracted to develop software to manage the inventory, sales, and arrangement of books within their store.

This system will serve the following use cases:

- **Users Administrators** can add new books to be tracked by providing ISBNs; the system will look up basic book details and allow the user to fill in the rest.

- Optionally, implementors may elect to provide automatic assistance in setting book prices. (extra credit)

- Optionally, implementors may elect to provide automatic assistance in identifying the book genre. (extra credit)

- **Users Administrators** will record wholesale purchases of books and reconcile store sales of books periodically.

- Users will be able to view a report of revenue, costs, and profits over time.

- The system will allow the tracking of book buybacks and provide statistics to help facilitate purchasing and buyback decisions.

- The system will include cover images as a part of book metadata and display them when appropriate.

- Users can use the system’s knowledge of book sizing to help lay out store shelves. (extra credit)
- Users will be able to perform a CSV bulk export of book information, and use a CSV bulk import to populate purchase orders, sales reconciliations, and book buybacks.

- IT operations staff will be able to restore the state of the system from a robust backup system using a clearly documented procedure.

- The system will support multiple users, with some being marked as administrators. Broadly speaking, administrators will be able to access “read-write” functionality, while normal users will have “read-only” access. This will allow regular employees to look up inventory while reserving management operations for authorized personnel.

- The system will be able to identify “related books” (multiple printings, editions, and/or translations of the same book).

- Users will be able to lay books out in a full bookcase to plan out highly visible displays, such as bestsellers and new releases.

- Administrators will be able to log unexpected changes in inventory (loss, theft, etc.).

- Book sales will be recorded automatically from Point Of Sale systems, obviating the need for sales reconciliations.

2 Definitions

1. Implementor: Refers to you, the software developer. Items described as “at the implementor’s discretion” or similar indicate free choices. However, “free choice” does not mean all choices are of equal merit; your overriding goal must be software quality.

2. Unique: Requirements may describe a given field (or combination of fields) as unique. This means that there may be at most one record with that value (or combination of values). Attempts to violate uniqueness should generate an error, unless otherwise specified in the requirements.


4. External database: A third party service that provides an API for looking up facts about a book, e.g. via its ISBN. The system will need to make use of one or more external databases to ascertain facts about books to be sold without having the user input every detail. Examples include Open Library and Google Books, but there are many more. Some are paid services; it is not necessary to use a paid service, but you are allowed to (on your own dime).
5. **Intrinsic property**: Aspects of a book that are inherent to the book itself and are generally found in an external database (title, author(s), etc.). Some intrinsic properties may be overridden by HB, e.g., to correct an error or omission in information from an external database.

6. **Extrinsic property**: An aspect of a book that fully controlled by HB (e.g., retail price).

7. **Monetary value**: An amount of money specified in United States Dollars (USD), e.g., $15.99. No fractional cents permitted.

8. **Genre**: Identifies the department in which this book will be displayed in store (e.g., “scifi”, “biography”, etc.). Defined by field:
   - Name: The name of the genre.

9. **Book**: A book eligible for sale by HB. Defined by fields:
   - Title: The title of the book. Short text field. Intrinsic and acquired from external database, not editable.
   - Author(s): The author(s) of the book. Implementors may store this as a single short text field capable of listing multiple authors (“author1, author2”) or as a tuple of discrete author strings ({“author1”, “author2”}). Intrinsic and acquired from external database, not editable.
   - Publisher: The publisher of the book. Short text field. Intrinsic and acquired from external database, not editable.
   - Publication year: The year the book was published. Intrinsic and acquired from external database, not editable.
   - Page count: The integer number of pages. Intrinsic and acquired from external database, if available. May be overridden or provided if absent from external database, or left unspecified.
   - Dimensions: The size of the book: a floating point number of inches for each of width, height, and thickness. Intrinsic and acquired from external database if available (converting units if necessary). May be overridden or provided if absent from external database, or left unspecified.
   - Retail price: A monetary value specifying the price that HB will charge customers for the book. Must be positive. Extrinsic, required.
   - Genre: A reference to an existing genre defined in the system (def 8). Extrinsic, required.
   - Cover: A web-viewable image file depicting the cover of the book. Ideally, should be suitable for full-screen display in a desktop browser. Intrinsic and acquired from external database if available. May be overridden or provided if absent from external database, or left unspecified.
10. **Vendor**: A wholesale bookseller. May be a publisher, printer, or distributor. Includes large publishers (e.g., Harper Collins), smaller companies (e.g., Space Wizard Science Fantasy), and even print-on-demand services (e.g., Lulu). Defined by fields:

- **Name**: The name of the vendor. Required.
- **Buyback rate**: The default pricing policy for book buyback from this vendor, expressed as a percentage of wholesale price (e.g., 35%, meaning the vendor will pay HB 35% of the wholesale unit price to buy back books from that vendor). Optional; omitting this value means that the vendor has no buyback program.

11. **Purchase order**: A wholesale purchase of one or more books from a vendor. Defined by fields:

- **Date**: The date of the purchase. Required.
- **User**: The user that created the purchase order. Required.
- **Vendor**: A reference to an existing vendor defined in the system (def 10). Required.
- **Purchase(s)**: A set of one or more invoice items, where each item is a tuple of:
  - **Book**: A reference to an existing book defined in the system (def 9). Required.
  - **Quantity**: The positive integer number of this book purchased. Required.
  - **Unit wholesale price**: A monetary value specifying how much was paid for the book per copy. Must be zero or positive. Required.

12. **Sales reconciliation**: An accounting record in which sales of books are logged by the system. Defined by fields:

- **Date**: The date of the reconciliation. Required.
- **Sale(s)**: A set of one or more sales totals, where each total is a tuple of:
  - **Book**: A reference to an existing book defined in the system (def 9). Required.
  - **Quantity**: The positive integer number of this book sold. Required.
  - **Unit retail price**: A monetary value specifying how much was received for the book per copy. This is the book’s retail price at the time of the reconciliation. Required.

13. **Inventory count**: The number of a given book that HB has in stock. This can be computed as the sum of purchase order acquisitions of a given book (def 11) minus the sum of sales of that book (def 12 29 for sales reconciliations records and def 22 for book buybacks). Adjusted by inventory corrections (def 26).

14. **Extra credit**: Some requirements are left up to the implementor’s discretion and are labeled “extra credit”. Naturally, you don’t have to do these. Further, you can choose when to do them: you will receive the awarded extra credit points in just the evolution in which the
feature first appears. However, some extra credit is time bounded to a particular evolution; this will be explicitly labeled. Exact values of extra credit are not given, but as with all credit, will be commensurate with the scope of work anticipated.

15. **Cost most-recent**: The wholesale cost of the book listed in the most recent purchase order. When used in a timespan-based report (such as the sales report), it’s based on the most recent purchase order up to and including the end-date of the report (but may go back earlier than the start-date of the report). When used in reference to a particular vendor (e.g., during creation of a buyback record), the cost most-recent is computed only with respect to that vendor. The rationale for this metric is that this is the cost most likely to be incurred in purchasing more copies of the book as well as the value most likely used in the computation of a buyback.

16. **Comma-Separated Values (CSV)**: A common data interchange format compatible with all modern spreadsheet software. The requirements call for various CSV import and export functions. All implementations should use identical CSV formats, but the exact specification of these formats is up to the consensus of the class. Each implementor group will nominate a member to a bulk import/export format committee to develop a proposed set of specifications to submit to HB (as embodied by the instructor).

17. **Shelf space**: A measure of horizontal distance on store shelves. For the purposes of shelf space estimation, when a book has unknown dimensions, calculations (defs 18 and 19) should assume 5 inches wide, 8 inches tall, and 0.8 inches thick.

18. **Spine-out**: Describes one way in which books may be placed on store shelves, with the spine facing out (like a library). Here, the shelf space occupied is the book thickness times the number of copies.

19. **Cover-out**: Describes one way in which books may be placed on store shelves, with the cover facing out (like a magazine stand). This is usually done for major new releases or other featured books. Here, the shelf space occupied is the book width, but multiple copies occupy finite shelf depth, which is 8 inches; the book thickness times the number of copies cannot exceed this.

20. **Last-month sales**: Total number of sales of a book in the last 30 days as reported in sales reconciliation records (but not buybacks).

21. **Days of supply**: The inventory count (def 13) of a book divided by its last-month sales (def 20) times 30. If last-month sales is zero, this is effectively infinite: it should be displayed as “(inf)” and sorted above numeric values.

22. **Book buyback**: Some vendors, especially those associated with large traditional publishers, allow booksellers such as HB to return unsold books for a percentage fraction of the wholesale cost, in a process called a buyback. A particular record of HB exercising a book buyback is defined by fields:
   - Date: The date of the buyback. Required.
   - User: The user that created the buyback record. Required.
23. **Best buyback price**: The highest price the book can fetch in a buyback. Given by finding the most recent purchase order of the book for each different vendor that HB bought it from and finding the maximum value of the vendor’s buyback rate times the wholesale price in the purchase order. For example, if vendor 1 most recently sold us the book for $10 and has a buyback rate of 45% (buyback unit price $4.50), that would be better than a vendor that sold it to us at $12 but has a buyback rate of only 20% (buyback unit price $2.40), thus the best buyback price would be $4.50. If no vendor can be found that has a buyback policy, then this is $0, simply displayed as “-”.

24. **User**: An account able to login and access the system. Defined by fields:

- Username: An alphanumeric user name. Required.
- Password: The password required to login; stored securely (req 1.4). Required.
- Administrator permission: A boolean flag indicating if this user has administrator privilege. Required.

25. **Administrator**: A user with administrator permission.

26. **Inventory correction**: Employees may, from time to time, take an inventory of actual books on hand, and if this differs from the system’s conception, log an inventory correction. This accounts for loss, theft, and accidental damage. Defined by fields:

- Date: The date of the correction. Required.
- User: The user that made the correction. Required.
- Book: A reference to an existing book defined in the system (def 9). Required.
- Adjustment: A positive or negative integer change to the inventory count of this book. Required.

27. **Point Of Sale (POS) system**: The common term for cash registers and associated equipment used for retail sales. The particular POS systems used by HB are older and only communicate with other equipment via RS-232 serial protocol.

28. **BookHook system**: A system designed to integrate with the POS systems via RS-232 serial in order to send transactions to an external system. The system was hacked together by the owner’s nephew in 2002 and, until recently, was pointed at a bit of PHP code to update
spreadsheets. However, because your software is displacing those spreadsheets, the time has come for you to integrate with BookHook directly. It works be listening to POS systems via serial and, when a sale is made, converting that sale into standards-compliant XML to be posted via HTTP(S) to a configurable URL. Unfortunately, no documentation survives except that provided by the tool itself. Due to potential line noise on RS-232 channels, reliability issues with the aging hardware, and bugs in the unsupported 20+ year old code, posts made by BookHook to your system may contain errors.

Meta-note: This system is designed to resemble the real antiquated and inscrutable code-bases that hold up many organizations. Each group has been assigned a separate instance at http://bookhook.colab.duke.edu:800X, where X is your group number. A “scratch” instance is also available at http://bookhook.colab.duke.edu:8000 for shared testing. The tool allows you to change the target URL of the sale POSTs, and has a “simulate” feature to post a new sale. In reality, I don’t actually have cash registers hooked up via RS-232, so the simulate feature will actually be the only way the tool reports sales to your system.

29. Sales record: An record of a single sale transaction from the POS systems as provided by BookHook (def 28). Defined to be the exact same fields as the old sales reconciliation (def 12); the difference is that sales records represent a single transaction with a single customer, whereas sales reconciliations were manually entered based on receipts from many such transactions.

30. Related book detection algorithm: An algorithm to attempt to automatically identify related books (def 9). There is a wide variety of possible approaches here, including using extended fields from your external database(s) or fuzzy matching on title. The bare minimum implementation is a whitespace-insensitive, case-insensitive title match. More advanced approaches may be implemented for extra credit; such an approach must be described and demonstrated in your feature guide if pursued.

3 Requirements

A note on requirements: No set of requirements is perfect, and that is certainly true here. I’m sure that contradictions, under-specified behavior, and unintended consequences will be revealed. Your overriding goal should be to produce a quality system; if you believe that goal would be better served if a requirement were altered or interpreted a certain way, ask about it, and get the conclusion in writing. The result may be a variance in a requirement for a specific team, or even modification of this requirements document for all teams. In short, if unsure, ask.

Some requirements have attached an informal tip, motivation, or example; these do not alter the requirements themselves, but are meant to answer likely questions about a requirement.

Note: If a requirement says that an administrator can do something and a sub-requirement simply says that the “user” will do part of that thing, then the word “user” in that context is simply a pronoun referring to the initiating administrator, not just any normal user.

1. Server

1.1. Your software must have a server that supports an arbitrary number of simultaneous users (even though access is governed by a single password). A web-based solution is preferred; thick client or mobile options are available with instructor pre-approval only.
1.2. During the install/setup process, a global access password is set and an administrator user called admin is configured. Users must have their accounts created by admin before being able to use the system (req 7).

1.3. A user accessing the system prior to logging in should be able to access nothing but a login prompt. Login is via a single global username and password.

1.4. The stored password must be kept in a secure manner (i.e., salted and hashed at minimum).

1.5. Users may change the global their own password using the customary two-matching-blinded-inputs approach commonly seen.

1.6. All communication between the clients and server must be encrypted.  
   Tip: For web-based solutions, this means using HTTPS.

1.7. The server must maintain state in a persistent fashion.  
   Tip: For web-based solutions, this just means using a database or similar.

1.8. **Pagination rule:** For all views which show a potentially unbounded number of records, the response time of the interface shall not depend on the quantity of records unless a full listing is explicitly requested by the user.  
   Tip: This implies some form of pagination so that only a finite number of records are retrieved at a time. Pagination can be explicit (page 1 of N) or implicit (infinite scrolling with auto-loading). The latter part of the requirement (“unless a full listing is requested”) implies a “show all” button or similar. Other UI solutions are likely also possible.

1.9. **Consistency rule:** A variety of cross-references are made by the system; the system must maintain internal consistency of these references in all cases. For example: if a genre book listing shows a given book, then the detail view for that book should show that same genre.

1.10. **Assisted selection:** A user input is said to be assisted if it is a user-selected reference to an existing record (e.g., book, vendor, etc.) where the UI provides a listing, inline search, autocomplete, and/or other means to allow easy and efficient selection. Unless otherwise specified in this document, all selections of an existing record should be assisted. In particular, for book selection (e.g., when logging a purchase order per req 3.7), a simple listing is insufficient (as it would be too big), so some form of search on title and/or ISBN must be incorporated (implementors may elect to make the search cover more fields than that at their discretion). One exception to the assisted selection requirement is data imported via CSV, such as line items for purchase orders (req 3.7.1).

1.11. **Image file handling:** Requirements relating to the cover art of books involve the acquisition, storage, and display of images. Images acquired from the user or an external database should be verified as valid image files of a web-accessible format (JPEG, PNG, GIF, WEBP); invalid files should be rejected. If coming from the user, an appropriate error should be shown. Image files, once committed to the system, should be stored by the system. Using URL references to images stored on unrelated third-party servers (“hot-linking”) is not permitted.

2. **Book record management**
2.1. **Book list**: Users will be able to view a table of books with most or all fields (publisher, publication year, page count, and dimensions may be hidden for space at the implementor’s discretion). The table will also include the computed values listed in req 2.1.5.

2.1.1. The view should be sortable by any of the displayed fields.

2.1.2. It should be possible to filter this view by selectable genre and by keyword search on book title, author, publisher, or ISBN (either 10 or 13).

2.1.3. Users should be able to navigate from this to a detail view for a book (see req 2.2).

2.1.4. For books that have cover art stored, it should be shown in thumbnail form in this view.

2.1.5. The listing should also include these computed fields:
   - Inventory count (def 13).
   - Shelf space used in inches (def 17), assuming spine-out placement (def 18). Books with an unknown thickness should have this value shown distinctively so it’s clear an estimate is being used.
   - Last-month sales (def 20).
   - Days of supply (def 21).
   - Best buyback price (def 23).
   - Number of related books (def 9).

2.1.6. From this view, it should be possible to generate a downloadable CSV export of the filtered data (ignoring pagination). The system shall provide easily understandable end-user documentation as to the CSV format.

2.2. **Book detail**: Users may view a detail view of a book showing all fields, and to request to modify the book from here (req 2.4). All computed values described in req 2.1.5 should also be shown. The cover art, if present, should be shown at a decently large size here (i.e., larger than a thumbnail).

2.2.1. A unified listing of purchase orders, sales reconciliations records, and book buybacks, and inventory corrections that include this book should be shown, sorted by date. The different types of transactions should be distinguished clearly, and shown with a running inventory total (e.g., “purchase order of 25 (25 on hand), sale of 5 (20 on hand), inventory correction of -2 (18 on hand”)”). A suitable level of detail should be provided (date, user, vendor (if applicable), price (if applicable), and quantity of copies). It should be possible to navigate from here to a detail view for a particular one of these items (reqs 3.6, 3.11, 3.17), excluding inventory corrections.

2.2.2. A listing of all related books should be shown with title, author, publisher, publication year, plus other metadata at the discretion of the implementor. It should be possible to navigate from one of these related book listings to the detail view for that related book.

2.3. **Book adding**: Users Administrators will be able to add books to the system as follows.

2.3.1. The user will input one or more ISBNs into a multiline text field. Dashes in the ISBN will be ignored, and ISBNs will be separated by newlines, tabs, spaces, and/or commas.
2.3.2. Upon submission, the system will consult one or more external databases to gather the books’ intrinsic properties. All fields associated with the book (as given in def 9) will be shown in a table format (with the cover in thumbnail format). Any duplicate books (i.e., those already in the system) will be shown distinctively, and can be edited here as if they were new books. This table need not follow the pagination rule (req 1.8).

2.3.3. Users will be able to view the intrinsic and extrinsic fields, modifying those identified as editable in def 9. The system will enforce constraints from def 9. For cover art, this means allowing the user to upload a new image or remove the currently associated image.

2.3.3.1. The system should use its related book detection algorithm (def 30) to identify related books. The number of these should be shown during this review, with the specific related book details visible by a popup or expandable view. As described in def 30, extra credit is available here by implementing a more sophisticated detection algorithm.

2.3.4. (Extra credit) Implementors may elect to provide a price suggestion based on an external pricing database.

2.3.5. (Extra credit) Implementors may elect to provide a genre suggestion. Because genres are user-configurable (req 2.7), this may be tricky. Possible solutions may involve library categorization number (e.g. Library of Congress Classification code), tagging system genres to tracked external database genres, or other approaches.

2.3.6. Upon successfully submitting, the new books (and edited duplicate books) will be committed to the system.

2.3.7. After this point, the system should not consult any external database for facts about a book that have been input here. In other words, this information should be stored by the system persistently.

2.4. Book modify: Users Administrators may modify a book to change editable fields. The system should enforce constraints as described in def 9. If req 2.3.4 or req 2.3.5 are implemented, they should be available here as well. For cover art, this means allowing the user to upload a new image or remove the currently associated image.

2.4.1. Inventory correction: The administrator should be allowed to modify the inventory count to any positive integer. In doing so, the old count should be shown alongside the new, as well as the delta being applied (e.g. “−4” or “+4”). A visible and clearly worded warning should be shown before saving this change. Saving such a change logs an inventory correction with today’s date.

2.5. Book delete: Users Administrators will be able to delete a book. This should be permitted only if the inventory count of the book is zero. Further, a highly visible confirmation dialog will be displayed first.

2.6. Genre list: Users will be able to view a table of genres showing the genre name and the number of books in that genre.

2.6.1. Users should be able to select a genre to navigate from this view to a book listing view for the selected genre (see req 2.1).

2.7. Genre create: Users Administrators will be able to add genres by name. If additional fields are needed to implement optional req 2.3.5, they should be provided here.
2.8. **Genre modify**: Users Administrators may rename a genre. If additional fields are needed to implement optional req 2.3.5, they should be modified here.

2.9. **Genre delete**: Users Administrators will be able to delete a genre only if that genre has no books associated with it, and only after a a highly visible confirmation dialog.

3. **Inventory and sales management**

3.1. **Vendor list**: Users will be able to view a sorted table of vendors showing name and buyback rate (if any).

3.2. **Vendor create**: Users Administrators will be able to add vendors by specifying vendor name and buyback rate (if any).

3.3. **Vendor modify**: Users Administrators may rename a vendor or change/remove its buyback rate.

3.4. **Vendor delete**: Users Administrators will be able to delete a vendor only if no purchase orders have been logged against it, and only after a a highly visible confirmation dialog.

3.5. **Purchase order list**: Users will be able to view a table of purchase orders showing the date, user, vendor, number of unique books, total number of books, and total cost.

3.5.1. The view should be sortable by any of the displayed fields.

3.5.2. Users should be able to navigate from this to a detail view for a purchase order (see req 3.6).

3.6. **Purchase order detail**: Users may view a detail view of a purchase order showing all fields, including a list of the line items (book, quantity, unit price, subtotal for that book), as well as a grand total cost. Users may request to modify the purchase order from here (req 3.8).

3.6.1. It should be possible to navigate from this view to the detail view for a particular listed book (req 2.2).

3.7. **Purchase order create**: Users Administrators will be able to log new purchase orders to the system by specifying a date (defaulting to today’s date), picking an existing vendor, and inputting one or more line items (book, quantity, wholesale unit price).

3.7.1. For the line items, the user may input this manually or via import from a CSV file. After CSV import, the user should be able to review, modify, add to, and delete from the imported line items as if they were manually entered before committing the purchase order. The system shall provide easily understandable end-user documentation as to the CSV format.

3.8. **Purchase order modify**: This operation should be rare. Users Administrators may modify a past purchase order to correct a prior mistake. A highly visible confirmation dialog will be displayed before committing any change. The system should enforce the constraint that inventory count for a book cannot go negative.

3.9. **Purchase order delete**: This operation should be rare. Users Administrators may delete a past purchase order if it was logged in error. A highly visible confirmation dialog will be displayed before doing so. The system should enforce the constraint that inventory count for a book cannot go negative.
3.10. **Sales reconciliation record list**: Users will be able to view a table of sales reconciliations showing the date, number of unique books, total number of books, and total revenue.

3.10.1. The view should be sortable by any of the displayed fields.

3.10.2. Users should be able to navigate from this to a detail view for a sales reconciliation (see req 3.11).

3.11. **Sales reconciliation record detail**: Users may view a detail view of a sales reconciliation showing all fields, including a list of the line items (book, quantity, retail price, subtotal for that book), as well as a grand total revenue. Users may request to modify the sales reconciliation from here (req 3.13).

3.11.1. It should be possible to navigate from this view to the detail view for a particular listed book (req 2.2).

3.12. **Sales reconciliation create**: Users will be able to log new sales reconciliations to the system by specifying a date (defaulting to today’s date) and inputting one or more line items (book, quantity, retail price). The system should enforce the constraint that inventory count for a book cannot go negative.

3.12.1. For the line items, the user may input this manually or via import from a CSV file. After CSV import, the user should be able to review, modify, add to, and delete from the imported line items as if they were manually entered before committing the sales reconciliation. The system shall provide easily understandable end-user documentation as to the CSV format.

3.13. **Sales reconciliation modify**: This operation should be rare. Users may modify a sales reconciliation to correct a prior mistake. A highly visible confirmation dialog will be displayed before committing any change. The system should enforce the constraint that inventory count for a book cannot go negative.

3.14. **Sales reconciliation record delete**: This operation should be rare. Administrators may delete a sales reconciliation record if it was logged in error. A highly visible confirmation dialog will be displayed before doing so.

3.15. **Sales report**: Users will be able to select a start and end date (inclusive), and generate a report of sales that includes the following components.

3.15.1. The revenues (summed from sales reconciliations and buybacks (shown separately)), costs (summed from purchase orders), and profits (the difference of the two) for each day within the time period, as well as a total revenue, cost, and profit for the full time period.

3.15.2. A table of the ten top selling books for the time period by quantity of copies sold, sorted in descending order. If fewer than 10 unique books were sold, a shorter list may be displayed. For each book, the quantity sold, total revenue, total cost most-recent (def 15), and total profit (the difference of the two) should be shown. Total revenue should be based on a sum of sales reconciliations for the period.
3.16. **Book buyback list**: Users will be able to view a table of book buybacks showing the date, **user**, vendor, number of unique books, total number of books, and total revenue.

3.16.1. The view should be sortable by any of the displayed fields.

3.16.2. Users should be able to navigate from this to a detail view for a buyback (see req 3.17).

3.17. **Book buyback detail**: Users may view a detail view of a book buyback showing all fields, including a list of the line items (book, quantity, buyback unit price, subtotal for that book), as well as a grand total revenue. Users may request to modify the buyback from here (req 3.19).

3.17.1. It should be possible to navigate from this view to the detail view for a particular listed book (req 2.2).

3.18. **Book buyback create**: Users **Administrators** will be able to log new buybacks to the system by specifying a date (defaulting to today’s date), picking an existing vendor, and inputting one or more line items (book, quantity, buyback unit price).

3.18.1. The user should only be allowed to choose a vendor that has a buyback policy.

3.18.2. For the line items, the user may input this manually or via import from a CSV file. After CSV import, the user should be able to review, modify, add to, and delete from the imported line items as if they were manually entered before committing the buyback. If line item(s) are in violation of the rules described herein, they should be marked clearly for correction/removal and an informative warning shown; data in violation should not be committed by the system. The system shall provide easily understandable end-user documentation as to the CSV format.

3.18.3. In selecting book line items, only books that have been sold by the chosen vendor (i.e., appear in a prior purchase order with the vendor) may be selected.

3.18.4. Upon selecting a given book, the buyback unit price should default to the price computed for the chosen vendor (i.e., the cost most-recent (def 15) for the vendor times the buyback rate). This value should be changable by the user. For CSV import, a blank buyback unit price should be replaced by the vendor computed default described above, but a filled-in buyback unit price should override the vendor computed default. In either case, the GUI should allow modification of these values before committing the buyback.

3.19. **Book buyback modify**: This operation should be rare. Users **Administrators** may modify a past buyback to correct a prior mistake. A highly visible confirmation dialog will be displayed before committing any change. The system should enforce the constraint that inventory count for a book cannot go negative.

3.20. **Book buyback delete**: This operation should be rare. Users **Administrators** may delete a past buyback if it was logged in error. A highly visible confirmation dialog will be displayed before doing so.

3.21. **BookHook integration**: The system will provide a URL endpoint that can be given to the BookHook system (def 28). BookHook will then be able to directly post sales records from the POS systems to this URL, thus replacing sales reconciliations. See BookHook itself for the XML schema that describes the import format.
3.21.1. Upon receiving a valid post from BookHook, a sales record with the provided data should be created. Because postings represent actual sales in real life, your system is not allowed to decline a sales record due to it making an inventory count negative. Your system should cope with this.

3.21.2. A valid post is one where the XML matches the given schema and the fields within match definitions provided in this document (i.e., ISBNs must be a valid ISBN 13 or ISBN 10, etc.).

3.21.3. If an ISBN referenced is not known to the system, that individual line item will be discarded, but recognized ISBNs in the same record will be kept. If all ISBNs for a sales record are unknown, then that sales record will be discarded.

3.21.4. The output provided by your system back to BookHook isn’t examined, but it should be text-based and clearly indicate success or a specific error.

3.21.5. Your system must be highly robust against malfunctions in BookHook. It should accept only valid input (per req 3.21.2). Invalid input should result in no change, and the response to BookHook should include an informative error message. In no situation should your system ever throw an unhandled exception in response to any request made by BookHook.

3.21.6. For security, your system should only accept postings from the IP address of BookHook (bookhook.colab.duke.edu) or other specific IP addresses identified for testing purposes. Other systems attempting to post should receive a clear error message in response.

4. Documentation

4.1. Developer guide: A document shall be provided which orients a new developer to how your system is constructed at a high level, what technologies are in use, how to configure a development/build environment, and how the database schema (or equivalent) is laid out.

4.2. Deployment guide: A document shall be provided which describes how to install your software entirely from scratch. It should start by describing the platform prerequisites (e.g., Linux distro, required packages, etc.), then mechanically describe every step to deploying your system to production readiness.

4.2.1. In addition to covering how to install the system with “stock” default data, the procedure to install the system from scratch using backed up data should also be included (i.e., disaster recovery). ¹

4.3. Feature guide: Optional. If an extra credit requirement is pursued, document its design, benefits, and a walkthrough of how to demonstrate it here. If you pursued extra credit in an earlier evolution, maintain its documentation, modifying it if needed.

4.4. Backup admin guide: A document shall be provided which explains the backup solution so that a system administrator unfamiliar with your software could configure it from scratch, restore the database to any given backup, and test a backup for validity. See req 6.

¹Moved 2022-03-01: This requirement was previously listed under the feature guide, which was a mistake.
5. **Store organization planner**

5.1. *(Extra credit in evolution 2)* **Shelf calculator:** To allow quick computation of possible store shelf layouts, a shelf calculator tool will be provided. *(This requirement is a stretch goal for evolution 2, but may appear for regular credit either as is or in a modified state in evolution 3.)*

5.1.1. In this tool, the user will be able to select one or more books. A given book may be chosen more than one time (e.g., to allow for a combination of spine-out and cover-out display).

5.1.2. For each book selected, the current inventory count will be shown, as well as an adjustable *display count* (how many copies will be placed on this shelf), which defaults to the inventory count. The display count can then be adjusted up or down irrespective of the inventory count.

5.1.3. For each book selected, the user can toggle between spine-out and cover-out display modes. When cover-out mode is selected, the display count should be capped at the number that can fit in a stack on the shelf (per def 19).

5.1.4. As the above selections are being adjusted, the tool will continually display the computed shelf space used by each book, as well as the total shelf space (def 17).

5.1.5. If a selected book has unknown dimensions, it should be shown distinctly and this fact made clear, so the user knows that the given space is an estimate based on default sizing only.

5.1.6. The user can exit the tool to return to the normal interface; no persistent state needs to be stored.

5.2. **Case designer:** Users will have access to a book case designer to help lay out front-of-store displays for new releases, best sellers, and special event tie-ins. This tool builds on the shelf calculator above (which should be maintained as a separate feature).

5.2.1. Upon entering the tool, the user will be able to start a new bookcase or load an existing one. Existing saved bookcases from all users will be available for selection, shown with their saved name, the user that created it, the user that last edited it, and the date/time of last edit.

5.2.2. After starting a new case or loading an existing one, the user will be able to (re)define the bookcase in terms of its width (default 60 inches) and number of shelves (default 7).

5.2.3. From there, users can select/remove books and set their display mode and display counts as in the shelf calculator (req 5.1).

5.2.4. The user will be able to arrange these books on a given shelf and to move them among shelves. As this is done, the shelf space used per shelf should be shown, and shelves that are overloaded (are attempting to use more shelf space than the width specified for the case) should be shown distinctively so this can be corrected.

5.2.5. Users may do the customary “save” and “save-as” actions to store the bookcase for later use. Saved bookcase names must be unique.

5.2.6. Implementors have broad freedom to design this UI element, and should do so with a focus on usability and efficiency.
6. **Backups**: You must deploy a backup solution for your system’s database.

   6.1. Backups shall be automatic and taken daily.
   6.2. Backups shall be kept with a staggered retention (7 daily backups, 4 weekly backups, 12 monthly backups).
   6.3. Backups must be stored on a separate system.
   6.4. The backup system must require separate credentials to access.
   6.5. The backup system should report on progress and alert on failure; this could be via email or another directed communication mechanism.
   6.6. The backup system may be built either out-of-band from the main software (e.g., a background database dump restored manually by a sysadmin) or in-band (e.g., the software itself exporting its database using internal automation).

7. **User management**

   7.1. **User create**: Administrators will be able to create user accounts, providing all fields specified in def 24. Password creation will use the customary two-matching-blinded-inputs approach commonly seen.
   7.2. **User list**: Administrators will be able to view a table of all users showing all fields (except password, of course). Users with administrator privilege should be clearly distinguished.
   7.3. **User modify**: Administrators should be able to give or revoke administrator status to user accounts other than their own. The admin account is special and cannot lose administrator privilege. Administrators may also reset a user’s password, again using the customary two-matching-blinded-inputs approach commonly seen.
   7.4. **User delete**: Administrators will be able to delete user accounts other than their own account and the admin account. A clear confirmation dialog should be shown first. Deleted users should still have their usernames shown appropriately on entities they created (e.g., purchase orders, etc.).

The following is a preview of a requirement that will be part of Evolution 4, shown early so you can be thinking about it.

- **Marketing video**

  1. Your company’s leadership has decided that it may be feasible to market this software to other booksellers. As your company is a small start-up, you do not have a formal marketing team, so your group has been asked to develop a 4-6 minute sales video to kickstart your effort.
  2. Points will be awarded for professionalism, succinctly capturing your value proposition, clearly differentiating from competitors, and overall attractiveness of visual aesthetic. Some extra credit points will be available; these will be awarded competitively.
  3. Submission of this component will be via YouTube link (either public or unlisted) submitted by a means to be announced separately.