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 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 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.

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
e 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 (convert-
ing 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.

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 field:

    • Name: The name of the vendor.

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

    • Date: The date of the purchase. 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).
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. Exact values of extra credit are not given, but as with all credit, will be commensurate with the scope of work anticipated.

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.

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.

   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 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 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).

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 inventory count.

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).

2.2. **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).

2.3. **Book adding**: Users 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 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. 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.

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 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.
2.5. **Book delete**: Users 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 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 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 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 list of vendor names.

3.2. **Vendor create**: Users will be able to add vendors by name.

3.3. **Vendor modify**: Users may rename a vendor.

3.4. **Vendor delete**: Users 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, 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).

3.7. **Purchase order create**: Users 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).

3.8. **Purchase order modify**: This operation should be rare. Users 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.\(^1\)

3.9. **Purchase order delete**: This operation should be rare. Users 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.\(^2\)

---

\(^1\) Added 2023-1-18

\(^2\) Added 2023-1-18
3.10. **Sales reconciliation 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 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).

3.12. **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 retail price will default to the one stored for the book at this time. The system should enforce the constraint that inventory count for a book cannot go negative.³

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.⁴

3.14. **Sales reconciliation delete:** This operation should be rare. Users may delete a sales reconciliation 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), 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, 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. Cost most-recent is the wholesale cost of the book listed in 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). The rationale is that this is the cost most likely to be incurred in purchasing more copies of the book.

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.

³ Added 2023-1-18
4 Added 2023-1-18
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.

4.3. **Feature guide**: Optional. If an extra credit requirement is pursued, document its design, benefits, and a walkthrough of how to demonstrate it here.

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 additional districts and transportation companies. 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.