UCL Student Library System

Authors: Ujan Anam, Rahul Kumar, James Loke, Ioan Thomas

Abstract: the proposal for Group 20 UCL Scenarios, an online library database management system to be used and run by UCL students.

Introduction

The idea being proposed relates to the software requirements of a student library system. A student library system would involve having dedicated physical areas and administrators who can take donations of used books from students who no longer need them, and loan them out to other students in the same way a library does.

The stakeholders for this system would be:

- UCL students, who cannot afford the more expensive textbooks or who want to recycle used books
- Former students, who no longer need textbooks they used
- Administrators, who ensure books are added to the system and physically manage the distribution and collection of books

The main idea for this is to increase the availability at which students can find books from reading lists or required for courses, especially when in short supply at university libraries, or more expensive to buy. A benefit of this is the implication of reduced waste from the production of books when they can instead be shared.

The primary goal of the software being developed will be so that people can easily view the available books online, and take these books out by making a digital request and collecting from the physical distribution centre.

Overview of Requirements

Any and all relevant jargon and terminology specific to this project can be found at the end under “Terminology” on page 7.

This system would involve trust and goodwill from all students as currently described, and has no means of managing the return quality of books after loaning has expired, from this it is clear the system will require:

- A system to keep track of all available books, including adding and removing available books and holding relevant book information.
- A system to request to take a book out on loan.
- A system to keep track of who currently has which book.
- The ability to search all available or currently loaned books.
- A system of tracking whether students have taken books out on loan previously, and if so, whether they returned them in the same condition and on time.

These basic requirements will be met with the following features:

- A credit system will be used.
- A database will be used to store all users on the system.
- A database will be used to store information on all available books.
- A database will be used to store information on the specific book item.

- A database will be used to store the relations between users and the books (items) they have borrowed
- The available books should be listed online, conveniently searchable by any of the listed fields for book information.

Key Information for Features

Credit System:

- All users will start with a base credit, which will be enough to borrow a lower quality book.
- When people donate books, administrators can add credit to their accounts in the database.
- When borrowing a book, returning the book within the time span given and in the same condition (determined by admin) as before borrowing, they will gain a small amount of credit.
- If the book is returned late, not at all, or in a worse condition they will have a proportionate deduction from their credit.
- If a user has 0 credit, they cannot borrow any books, depending on how much credit a user has above 0, they can borrow multiple books at once with high enough credit, the number and considered quality of books available to them is dependent on their credit.
- The duration a user can borrow a book for will also be dependent on their credit.

User Database:

- Users will be uniquely distinguished by their UCL email, and using the UCL API to provide login security to the system.
- Each entry into the database must contain a UCL email and an associated credit score.

Book Information Database:

- Must include the ISBN (unique ID), author(s), and publication date.
- May be outsourced to other book information databases for other information including descriptions and more information.

Book Item Database:

- The book items will include a uniquely generated book ID as the key, a current condition, and the book's ISBN to relate to the relevant entry in the book information database table.

User-Book Relationship Database:

- Includes the user's email, and the book item's ID, and a start date for the borrowing, these will ensure that even when a user borrows the same book multiple times each distinct event of borrowing will be logged separately.
- Additionally a field for the end date at which the book must be returned will be held.
- The end date should be updateable so a user can request an extension on the borrowing from the admins, and this will be dependent on their credit score and may require an onsite check on the book's condition before approval.

Search System:

- Books should be searchable by availability, i.e. whether they are currently loaned or available to loan.
- Books should be searchable by ISBN, titles, or authors.
- Books may be searchable by topic area, fiction/non-fiction, or publication date.

UI:

- The main experience of interacting with the library system will be website based.

Legal and Ethical Considerations

At its core, the UCL Student Library System has goals such as broadening access to information for students, promoting the sustainability of UCL, and reducing the financial burden of educational material.

The project primarily aims to achieve this by allowing students the opportunity to receive pre-owned books from other students for no charge. However, the project must consider legal and ethical issues related to the concept.

The UCL Student Library System will be storing user data which may lead to the rise of some ethical issues. We will have to adhere to the GDPR and Data Protection Act 2018[1]. To handle personal data transparently, users will have to approve a data sharing agreement[2] before they are able to borrow any books. User authentication will be handled through the UCL API which means that an account will not be created directly on the UCL Student Library System website. To ensure that user data is handled responsibly, UCL Student Library System will only request for necessary data from the API such as a user's UCL email.

UCL Student Library System must not be used to spread prohibited books. Books which encourage violence, promote discriminatory ideologies, or contain misinformation are some examples of books which should not be allowed to be listed on the website. To mitigate this risk, UCL Student Library System's administration team will have to regularly review book listings to ensure that the books adhere to the ethical guidelines and any books are not defaced with offensive content. Moreover, users who return defaced books will receive a credit deduction, warnings or suspension from the administration team.

[1] Data Protection, UK Government, https://www.gov.uk/data-protection
[2] Data Sharing Agreements, Information Commissioner's Office, https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/data-sharing-a-code-of-practice/data-sharing-agreements

Technical Components

Component	Technical Tools For Implementing	Justification/Reasoning
Backend	Python and Django	Python is a widely used language which will work well with the short timeframe for this project by providing extensive libraries and testing facilities for code. Django is an extensive and well-documented web development framework for Python which provides tools for SQL integration and a REST API.
Database system	MySQL	Provides scalability, and integrates with Django well. We are using an SQL database instead of a no-SQL database because this project is relatively small scale and will not require horizontal scaling capabilities. Group members are also more familiar with SQL than no SQL.
Search system		Will be handled by SQL queries on the database.
Account system	UCL API[3]	Transfers the responsibility of user security to UCL, and provides an easy way to both verify user identities and allow users to easily gain access to the system.
Website UI	ReactJS	ReactJS is a commonly used JavaScript library for frontend web development and is extensively documented. The framework provides built-in tools that facilitate the development of inclusive user interfaces, ensuring compatibility with screen readers, keyboard navigation, and other assistive technologies. It also provides flexibility allows across desktops and mobile phones, ensuring a consistent and accessible experience for all users.

Primary Algorithms and Data Structures

SQL Queries:

- SQL databases use optimisers to make search queries more efficient. These optimisers do not always choose the most efficient query plan for any given query.
- SQL databases use cost-based optimisers that:
    - Estimate query cost (CPU, disk I/O, memory).
    - Choose the cheapest execution plan.
- Different database management systems (DBMS) use different optimisers:
    - MySQL - Query Optimiser.
    - PostgreSQL - Cost-Based Optimiser.
    - SQL Server - Query Execution Engine.
- If a column used in the WHERE clause (e.g., name) has an index, the search becomes much faster. Common types of indexes include:

[3] UCL API, https://uclapi.com/

    - B-Tree Index: Most commonly used, helps with range-based and equality searches.
    - Hash Index: Good for exact matches but not for range searches.
    - Full-Text Index: Optimised for searching text (e.g., LIKE and MATCH AGAINST queries).
- The SQL engine determines an execution plan, which might involve:
    - Full Table Scan (if no index is present).
    - Index Scan (if an index covers part of the query).
    - Index Seek (directly retrieving relevant rows using an index).
    - Join Algorithms (e.g., Nested Loop, Hash Join, Merge Join for multi-table queries).
- SQL supports different join methods:
    - Nested Loop Join, which is best for small datasets.
    - Merge Join, which is efficient for sorted, indexed data.
    - Hash Join, which is used when no indexes exist.
- The database uses a Cost-Based Optimiser to determine the most efficient plan. Once the relevant rows are found:
    - If there's an ORDER BY, SQL sorts the data.
    - If there's a LIMIT, SQL returns only a subset.
    - If the query involves joins, aggregations, or subqueries, additional operations occur.
- SQL queries for modern relational DBMS often nest several layers of SPJ (Select-Project-Join) blocks using groups of BY, EXISTS, and NOT EXISTS operators. In some cases these nested queries can be flattened into a SPJ query, but not always. Query plans for these nested queries may be chosen by using the dynamic programming algorithm used for join ordering, but there can be a huge escalation in query optimisation time.[4]

Data Structures:

- B-Trees and B+ Trees – Used in indexing to allow fast range and point queries.
- Hash Tables – Used in hash indexes for quick lookups.
- Heaps – Unordered storage for tables without clustered indexes.
- Linked Lists – Used in certain indexing and query execution structures.
- Tries (Prefix Trees) – Sometimes used for indexing textual or hierarchical data.
- Bitmaps – Used in bitmap indexes to speed up queries with low-cardinality columns.
- Graphs – Underlying structure for relational connections (foreign keys, joins).
- Bloom Filters – Used in some databases (e.g., PostgreSQL) to optimise query execution.

Data Storage:

- Tables – The primary data storage structure in SQL, consisting of rows and columns.
- Indexes – Speed up queries by providing quick lookups (e.g., B-Trees, Hash Indexes).
- Views – Virtual tables based on SQL queries, storing no data but simplifying access.
- Materialised Views – Stored query results that can be refreshed periodically.
- Partitions – Large tables split into smaller, manageable segments.

[4] "MySQL :: MySQL 8.0 Reference Manual :: 8 Optimization," https://dev.mysql.com/doc/refman/8.0/en/optimization.html

Development Plan and Feasibility

Week 1:

The first activity which must be completed will be a rough layout of the website UI, which will make it easier to debug the database and search functionality.

An activity which must be completed parallel to the first will be the backend code which establishes connection with an SQL server, and implements a database which meets the outlined criterion.

Week 2: The next activity will then be connecting the basic frontend with the basic backend through REST APIs, and establishing that the frontend can be used to make book borrowing requests, send search queries, and add and remove book items.

Week 3:

Next, a trivial system for user and administrator accounts will be established. This includes the features of users being able to search for books and request to borrow books. Additionally, administrators having the capabilities of adding (with an associated book condition and ISBN) and removing book items from the listing, updating the book item's condition, and updating a user's credit score.

The next step is to integrate the user account system with the UCL API, once done this will provide the fundamentally required features for the system to work.

Week 4/5: The final stage will be code cleanup and refactoring, documenting design, unit testing, and getting potential user feedback which can be acted on to make final changes to the UI.

Version control and integration will be done via GitHub.

Feasibility:

- From the development plan, there is a clear estimation that a group of four first year software engineers should able to complete this project given five weeks with around five hours per person's work a week, obviously communication overhead will be included in this timeframe.
- The technical details and development plan should show that the complexity of this project is sufficient to take this amount of time and demonstrate the expected degree of work and understanding for this assignment.

Diagrams

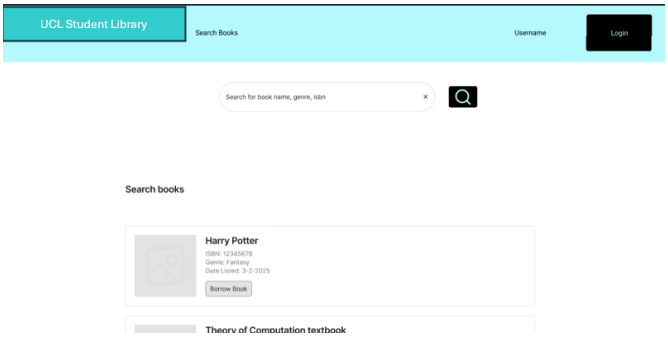

Figure 1. Rough Website Wireframe.

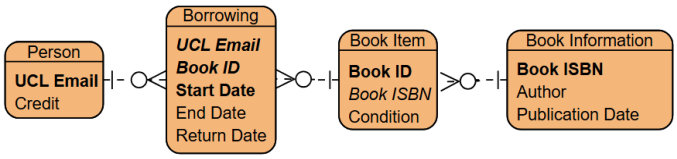

Figure 2. Preliminary Database Design, Entity-Relationship Diagram.

Future Additions and Expansion

As a team, we hope this project will be able to be expanded in the future to include:

- A system for exchanging books, i.e. a decentralised system for borrowing books where individual users can loan books they own and manage the timespan and condition books are kept and returned in.
- A system for exchanging other resources, such as physical hardware for engineering students, stationary for any students, or materials such as paint and other art based equipment.
- A system for purchasing books for long term access, especially if users want unrestricted use of books.
- Sharing of module and course notes, which may reward users with credit depending on how people may rate the quality and usefulness of notes.
- A feature allowing book reports and reviews, along with descriptions of course relevance or a reading list system.

Terminology

Onsite, or any reference to physical location, will refer to the location where the administrators are storing and distributing books from.

Stakeholders:

- User – any UCL student who is looking to borrow a book from the library.
- Donator – any UCL student who is giving books to the library which can be loaned.
- Administrator – ‘admin’, someone who has the ability to manage the available books, the credit scores of users, and view all loan requests.

Loaned/borrowed book – a book which has been given to a user by the library, and as such loaning/borrowing a book is providing that user with a requested book for a finite period of time.

Listing - referring to the books made available and searchable to borrow.

Start date - when the book was collected to be borrowed, a static datum.

End date - when the book must be returned by, a mutable datum.

Return date – when the book was (if ever) returned by the user who is currently borrowing it.

Condition of a book – how tattered and frayed the pages are, whether any stains exist, whether it is bound properly, whether the cover is intact. The level of condition will be measured in discrete units which will be decided upon at a later date.

Unrelated to Proposal Content

ENGF0034_24-25 » Forums » Discussion forum » if one were to include cat pictures in the appendix, is there any chance of being marked down?

Re: if one were to include cat pictures in the appendix, is there any chance of being marked down?

by Yuzuko Nakamura - Friday, 14 February 2025, 12:27 PM

As long as you make it clear that there's no project-relevant information past a certain point, and that the inclusion of cat pictures doesn't make marking more difficult (e.g. making the PDF file too big, slow to load, etc.), it should be fine.

Yes, references, title page, table of contents (if you choose to include those; they are not needed) do not count toward the page limit.

Show parent | Reply

See this post in context

Figure 3. Justification.

Figure 4. Being a cutie patootie as a puppy.

Figure 5. Sitting on his chair.

Figure 6. Under the sofa.