Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

AI Powered Smart Library Management System Using Computer Vision: Barcode Recognition, Facial Authentication, Voice Search and Hybrid Book Recommendation in a Browser Based Platform

Nnaemeka Ugwumba

nnaemeka_ugwumba@uniport.edu.ng

University of Port Harcourt https://orcid.org/0009-0000-2493-9846

Clement Barisua Ndeekor

clement.ndeekor@uniport.edu.ng

University of Port Harcourt, Port Harcourt, Nigeria

Okoro Nnaemeka Michael

University of Port Harcourt, Port Harcourt, Nigeria

Research Article

Keywords: smart library management system, artificial intelligence, computer vision, barcode recognition, ISBN scanning, facial recognition, FaceAPI.js, voice search, Web Speech API, book recommendation system, collaborative filtering, hybrid recommender, heat map, demand forecasting, academic library, Nigeria

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11823993/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

AI Powered Smart Library Management System Using Computer Vision: Barcode Recognition, Facial Authentication, Voice Search and Hybrid Book Recommendation in a Browser Based Platform

Clement Barisua Ndeekor
clement.ndeekor@uniport.edu.ng
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Nnaemeka Kingsley Ugwumba
nnaemeka_ugwumba@uniport.edu.ng
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Okoro Nnaemeka Michael
okoronnaemekamichael@gmail.com
Department of Computer Science, Faculty of Computing
University of Port Harcourt, Port Harcourt, Nigeria

Abstract

Most academic libraries in Nigeria still lend books with paper registers or basic catalogue software, which makes checkout slow, gives members no guidance on what to read next and gives librarians little evidence for buying decisions. This paper presents an AI powered smart library management system that runs entirely in the web browser. It combines computer vision barcode scanning of ISBN codes for checkout, facial recognition with 128 dimensional face descriptors for member authentication, voice search through the Web Speech API, a hybrid book recommendation engine, a category heat map and a demand forecasting module, with all data kept on the client. The system was built with HTML5, JavaScript, FaceAPI.js on TensorFlow.js, Html5Qrcode and Chart.js, and passed ten unit tests covering cataloguing, membership, borrowing, returns, voice search, recommendation, heat maps, face registration, face login and barcode scanning; pages loaded in 1.2 seconds and voice queries returned in 1.5 seconds. Because functional tests do not show how well the AI parts work, we evaluated two of them further. In a controlled study of 150 synthetic ISBN barcodes under simulated camera conditions, the standard decoder read every code in normal light, with perspective tilt up to 30 percent and rotation up to 20 degrees, but failed completely at 12 percent of normal light, at a blur of 2 pixels and at rotations of 30 to 60 degrees; a simple enhancement step (median filtering, contrast stretching and rotated retries) restored decoding to 99.3 percent at 8 percent light, 98.7 percent at a blur of 2 pixels and at least 98.7 percent at every rotation, at a median cost of a few milliseconds in good conditions. In an offline evaluation on 4.0 million interactions between 53,120 readers and the 2,000 most read titles of the goodbooks 10k dataset, item to item collaborative filtering reached a precision of 33.6 percent among four suggestions, against 12.8 percent for popularity and 15.1 percent for author and tag matching; a hybrid with equal weights, as first implemented, fell to 26.6 percent, while weights tuned on separate readers reached 33.8 percent with a hit rate of 70.7 percent. We discuss the privacy of face data under the Nigeria Data Protection Act 2023, the need for liveness detection, and the steps needed to move the system from a single browser to a shared library database.

Keywords: smart library management system, artificial intelligence, computer vision, barcode recognition, ISBN scanning, facial recognition, FaceAPI.js, voice search, Web Speech API, book recommendation system, collaborative filtering, hybrid recommender, heat map, demand forecasting, academic library, Nigeria

1 Introduction

A university library lends books, keeps track of who has them and helps readers find what they need. In many Nigerian institutions these tasks are still done with paper registers, card catalogues or basic catalogue software. A member queues at the desk, shows an identity card, and waits while the librarian writes the title, accession number and due date. Members who cannot find a book must ask a librarian or browse shelves, and librarians decide what to buy with little evidence of what is actually read. Integrated library systems such as Koha and Evergreen automate cataloguing and circulation, and commercial platforms add analytics, but none of them offers camera based checkout, face login, voice search or personalised suggestions out of the box, and commercial licences are beyond the budgets of most Nigerian libraries.

Artificial intelligence offers ways to close each of these gaps [1, 4]. A camera can read the barcode printed on every modern book. A face recognition model can confirm who is borrowing without a card. Speech recognition lets a reader search by speaking. A recommendation engine can learn from borrowing histories which titles go together. These technologies are now available as free JavaScript libraries that run inside an ordinary web browser, so a library can use them without servers or licence fees.

This paper describes such a system, built at the University of Port Harcourt, and evaluates its two central AI components. We ask four questions.

1. How can computer vision, facial recognition, voice search, recommendation and analytics be

combined in a library system that runs entirely in the browser?

2. Does the implemented system meet its functional and performance requirements?

3. Under which camera conditions does barcode based checkout fail, and can simple image processing make it more reliable?

4. How accurate is the hybrid recommendation approach compared with its parts, when tested on real borrowing data at the scale of a university library?

Our contributions are:

1. the design and implementation of a browser based smart library system that integrates ISBN barcode scanning, face login with 128 dimensional descriptors, voice search, hybrid recommendation, a category heat map and demand forecasting, with no server and no licence cost;

2. a functional and performance evaluation of the system with ten unit tests and load and response time measurements;

3. a controlled reliability study of camera barcode decoding over six capture conditions and six realistic library scenarios, with a low cost enhancement step that restores decoding in dim light, mild blur and rotation;

4. an offline evaluation of popularity, content based, collaborative and hybrid recommendation on 4.0 million real reading records, including the effect of the hybrid weights and of members with very short histories;

5. an analysis of the privacy and security of browser based face recognition under the Nigeria Data Protection Act 2023, with concrete recommendations.

Section 2 reviews related work. Section 3 presents the system design and Section 4 its implementation and testing. Section 5 describes the two evaluation studies and Section 6 their results. Section 7 discusses the findings and Section 8 concludes.

2 Related Work

2.1 Library systems and technology acceptance

Library automation has moved from card catalogues and paper ledgers to automated circulation in the 1970s and 1980s, integrated library systems in the 1990s, web based catalogues in the 2000s and, more recently, intelligent services. Open source systems such as Koha and Evergreen manage cataloguing, circulation, acquisitions and membership and are free to use, but they are complex to install and maintain. Commercial platforms such as LibSys, Ex Libris Alma and Innovative Interfaces Sierra add advanced analytics and cloud hosting at high cost. Table 1 compares them with the system described here on the features that matter for this study.

Table 1. Feature comparison of library management systems (Y: present, N: absent)

Feature	Koha	Evergreen	LibSys	Alma	Sierra	Ours
Open source	Y	Y	N	N	N	Y
Web based	Y	Y	Y	Y	Y	Y
Cloud hosted	N	N	N	Y	N	N
Cataloguing	Y	Y	Y	Y	Y	Y
Circulation	Y	Y	Y	Y	Y	Y
Analytics	Basic	Basic	Adv.	Adv.	Adv.	Adv.
Recommendation	N	N	N	N	N	Y
Voice search	N	N	N	N	N	Y
Face recognition	N	N	N	N	N	Y
Computer vision	N	N	N	N	N	Y
Demand forecast	N	N	N	N	N	Y
Cost	Free	Free	High	V. high	High	Free

Table 1 shows that the established systems cover the core library functions well, but none of them offers the AI features listed in its lower half, and the commercial ones carry high licence costs. The analytics of our system are marked advanced because of its heat map and forecasting modules, although they work on far less data than the commercial platforms.

New technology is adopted only if people accept it. The Technology Acceptance Model [6] explains use through perceived usefulness and perceived ease of use, and the Unified Theory of Acceptance and Use of Technology [28] adds social influence and facilitating conditions. Both guided our design: each AI feature had to save visible time (usefulness) and work with a single click or spoken phrase (ease of use). Information retrieval theory [14] frames search as an interaction between the reader’s need and the system, and Norman’s principles of user centred design [20], such as visible state and immediate feedback, shaped the interface.

2.2 AI in libraries

Cox et al. [4] interviewed library leaders and found that they expected AI to change search and discovery, the description of collections and the support of research, while raising concerns about privacy and skills. Asemi et al. [1] reviewed expert systems, AI and robots in libraries and identified cataloguing, reference services, recommendation and collection management as the main applications, with cost, data quality, expertise and bias as the main barriers. Few of the systems they reviewed were evaluated with measured accuracy, which is the gap this paper addresses.

2.3 Computer vision for book identification

Books can be identified from images in several ways: classifying cover or spine images, detecting books on shelves with object detectors such as YOLO [22] or SSD [19], reading titles with optical character recognition [27], or decoding the printed barcode. Since about 2007 the ISBN of almost every book has been printed as a 13 digit EAN barcode [15], so barcode decoding gives an exact identifier with very little computation. Camera based decoding is harder than laser scanning because of blur, low resolution and uneven light; Gallo and Manduchi [8] showed that phone cameras need methods that tolerate

blur. The open source ZXing decoder [32], on which Html5Qrcode is built, is the most widely used camera barcode reader on the web.

2.4 Facial recognition

Modern face recognition maps a face image to a fixed length vector, so that images of the same person lie close together and images of different people lie far apart [25, 30]. The FaceAPI.js library used here runs a ResNet based network [10] that produces a 128 dimensional descriptor, adapted from the face recognition model of the dlib toolkit [17], whose author reports 99.38 percent accuracy on the Labeled Faces in the Wild benchmark [12]. It runs on TensorFlow.js [26] with lightweight detectors based on MobileNet [11]. Two concerns are well documented. Accuracy differs across demographic groups, with higher error rates for darker skinned and female faces in many systems [2, 9], and a printed photo or a video can deceive a system that lacks presentation attack detection [21].

2.5 Recommendation systems

Recommendation methods fall into content based filtering, which suggests items similar to those a user liked; collaborative filtering, which uses the behaviour of other users; knowledge based methods; and hybrids [3, 23]. Item to item collaborative filtering [18, 24] computes how often two items are consumed by the same people and recommends the items most similar to a user's history; it scales well and explains itself ("readers who borrowed this also borrowed"). Cremonesi et al. [5] showed that top N accuracy measures such as precision and recall at a cut off, rather than rating error, should be used to compare recommenders, and that popularity is a strong baseline that complex methods do not always beat.

2.6 Voice search, analytics and forecasting

The Web Speech API [29] gives browsers access to speech recognition without extra software, which makes spoken search possible on any supported browser, although accuracy depends on noise and accent. Heat maps show the intensity of use of library sections or subjects at a glance, and borrowing data can be forecast with moving averages or regression [13] to guide acquisitions.

2.7 Research gap

Existing library systems lack AI features, and AI library studies seldom measure how well their AI components work under realistic conditions. This paper combines the features in one free system and measures the two that decide whether members will trust it: whether the camera reads the book, and whether the suggestions are relevant.

3 System Design

3.1 Methodology and requirements

The system was developed with an Agile process (Figure 1): six sprints of one to two weeks over twelve weeks, each passing through planning, design, development and testing, with feedback from users carried into the next sprint. This allowed features that depend on browser behaviour, such as microphone and camera access, to be tested early and adjusted.

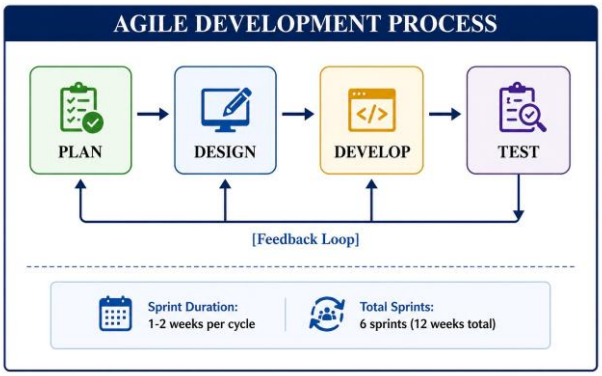


Figure 3.1: Agile development process adopted for the project.

Figure 1. Agile development process: six sprints of planning, design, development and testing with a feedback loop.

Figure 1 also shows why the iterative approach mattered here. Problems such as premature speech timeouts and slow loading of face models (Section 4.4) appeared only when the features ran on real devices, and the sprint structure let them be fixed before the next feature was added.

The functional requirements fall into nine groups: book management (add, view, edit, delete, track status), member management, borrowing and returns with due dates and a transaction log, computer vision book detection, facial recognition, recommendation, voice search, heat maps and demand forecasting, plus local data persistence. The nonfunctional requirements set response within 3 seconds, an intuitive interface, consent based face capture, reliable operation, support for all modern browsers, capacity for 2,000 books and 500 members, modular code and voice access for users with disabilities. The minimum hardware is an Intel Core i5 class processor, 8 GB of memory, a 720p webcam and a microphone; the software stack is listed in Table 2.

Table 2. Development tools and technologies

Purpose	Technology
Structure and style	HTML5, CSS3 (Flexbox and Grid)
Application logic	JavaScript (ES6)
Face recognition	FaceAPI.js on TensorFlow.js
Barcode scanning	Html5Qrcode (ZXing)
Voice recognition	Web Speech API
Charts and heat map	Chart.js
Data storage	browser LocalStorage
Tools and hosting	Visual Studio Code, Git, Netlify

Table 3 lists the nonfunctional requirements with their priorities. Performance, usability, security and reliability were rated high because a desk system that is slow, confusing, insecure or unstable would be abandoned in favour of the paper register, which is the practical meaning of the acceptance models discussed in Section 2.

Table 3. Nonfunctional requirements and priorities

Requirement	Description	Priority
Performance	respond to user actions within 3 seconds	High
Usability	intuitive interface that is easy to navigate	High
Security	secure facial recognition authentication	High
Reliability	operate without crashes or data loss	High
Portability	run on all modern web browsers	Medium
Scalability	support up to 2,000 books and 500 members	Medium
Maintainability	modular, documented code	Medium
Accessibility	voice search for users with disabilities	Medium

3.2 Architecture

Figure 2 shows the architecture. It has five layers, all inside the browser. The presentation layer holds the dashboard, catalogue, member, borrowing, report and settings pages. The application logic layer manages books, members, borrowing and search. The AI module layer contains the face recognition, recommendation, voice, heat map and forecasting modules. The data layer stores books, members and borrowing history as JSON in LocalStorage. The browser API layer gives access to the camera through getUserMedia, to the microphone through the Web Speech API and to drawing through the Canvas element.

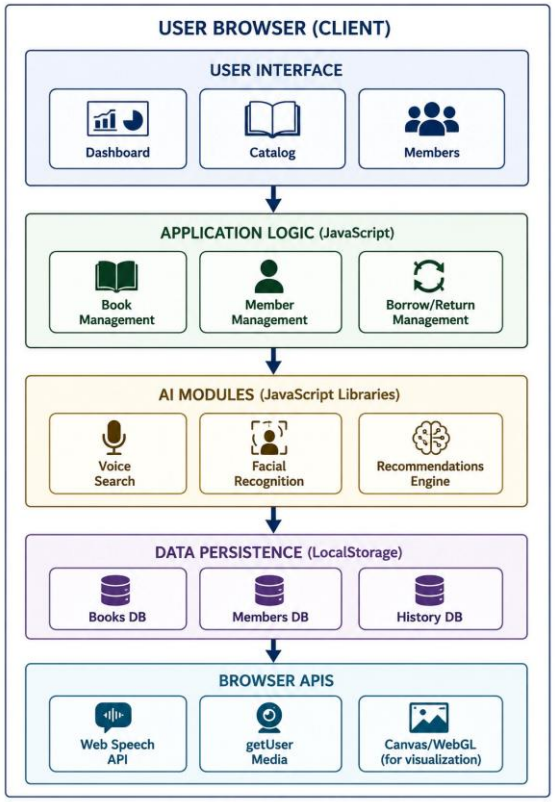


Figure 3.2: *High-level system architecture diagram.*

Figure 2. High level architecture: user interface, application logic, AI modules, data persistence and browser APIs, all running in the client.

A client only architecture, shown in Figure 2, was chosen for cost, privacy and simplicity. No server is needed, so the system can be hosted for free on a static host; face descriptors never leave the device; and the system works offline once loaded. The price of this choice is that each browser holds its own copy of the data, which limits the system to a single desk and to the five to ten megabytes that browsers allow for LocalStorage. Section 7 returns to this.

3.3 Use cases and data flow

Three actors use the system (Figure 3). The librarian manages books and members, processes borrowing and returns and views analytics. The member logs in by face, searches by voice or text, receives recommendations and borrows or returns books. A guest can browse the catalogue. Book detection by camera is triggered by either the librarian or the member.

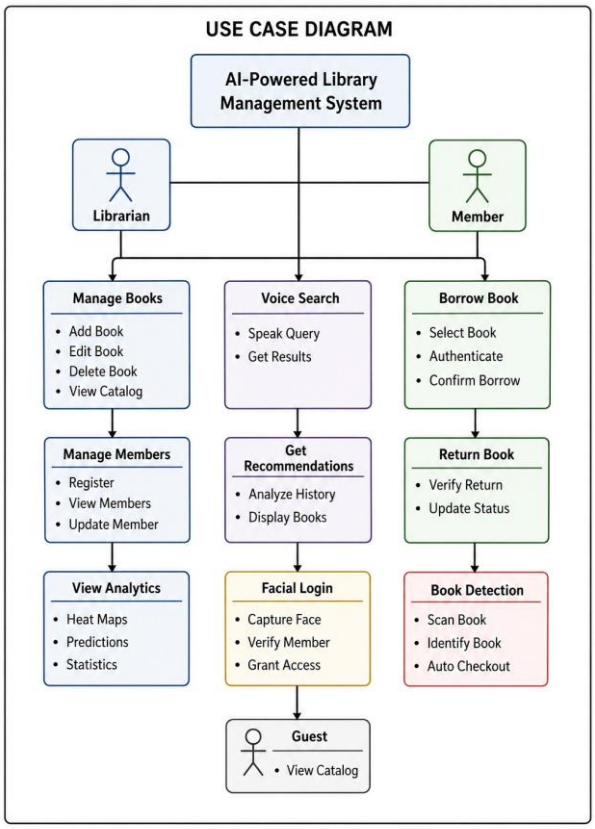


Figure 3.3: *Use case diagram showing system actors and their interactions.*

Figure 3. Use case diagram showing the librarian, member and guest actors and their use cases.

The use case diagram in Figure 3 shows that the AI features attach to existing library tasks rather than forming a separate tool: face login is a way to identify the borrower, barcode detection a way to select the book, and recommendations appear where members already look for books. Figure 4 shows the level 1 data flow. Five processes (book management, borrow and return management, catalogue and search, AI services and data persistence) exchange data with the three actors and with the book, member and history stores. The AI services process reads from all three stores, which is why the quality of the borrowing history directly limits the quality of recommendations and forecasts.

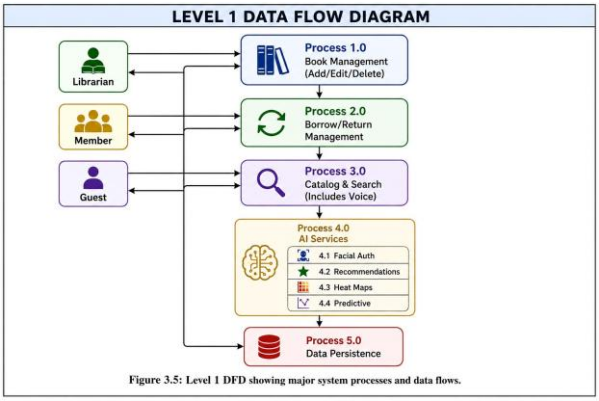


Figure 4. Level 1 data flow diagram: major processes, actors and data stores.

3.4 Data design

Figure 5 shows the three entities. A Book has an identifier, title, author, year, status, ISBN, category, description, the member who borrowed it and the borrow and due dates. A Member has an identifier, name, an optional face descriptor and a join date. A BorrowHistory record links a member to a book with an action (borrowed or returned) and a timestamp. One member may borrow many books, and each borrowing creates history records.

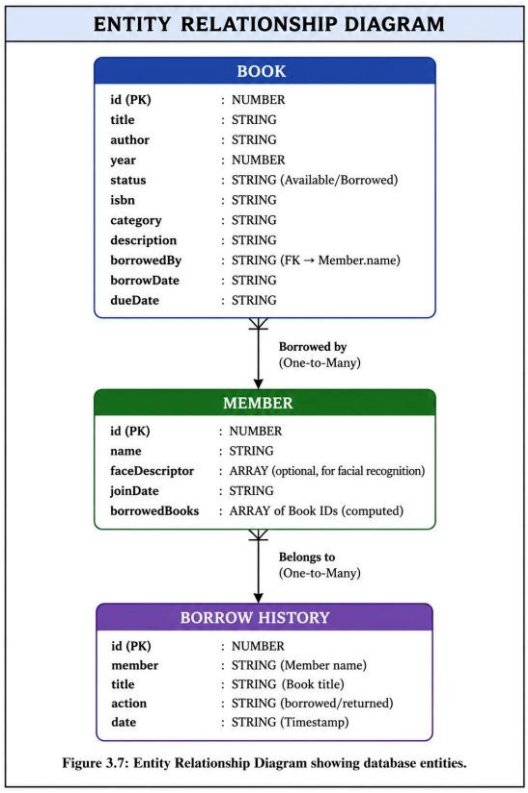


Figure 5. Entity relationship diagram of the Book, Member and BorrowHistory entities.

Two details of Figure 5 matter for the AI modules. The face descriptor is stored as an array of 128 numbers inside the Member record, so anyone with access to the browser profile can read it. And the history records carry a timestamp for every action, which is the input the recommendation and forecasting modules need.

Table 4 gives the data dictionary of the Book entity, the central record of the system. The ISBN field is what the barcode scanner matches, and the status, borrower and date fields are what the circulation, heat map and forecasting modules read.

Table 4. Data dictionary of the Book entity

Field	Type	Description
id	integer	primary key
title	string (255)	book title
author	string (255)	author name
year	integer	year of publication
status	string (20)	available or borrowed
isbn	string (50)	ISBN, matched by the scanner
category	string (100)	subject category
description	text	short description
borrowedBy	string (255)	name of current borrower
borrowDate	string (20)	date borrowed
dueDate	string (20)	date due, 14 days later

3.5 AI module design

Figure 6 shows the computer vision process for book recognition: camera input, preprocessing (noise reduction and normalisation), feature extraction, classification or detection, optical character or barcode decoding, and matching against the book database, ending with an identified book and a completed transaction. The implemented system follows the barcode branch of this process, because the ISBN barcode gives an exact match with little computation.

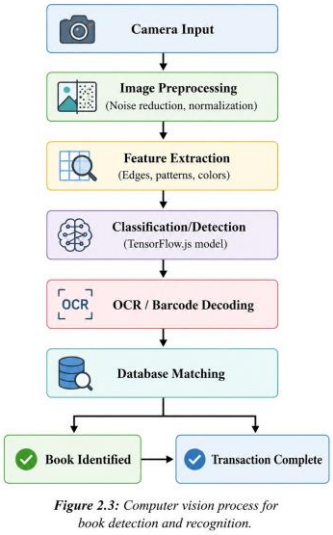


Figure 6. Computer vision process for book detection and recognition.

Figure 7 shows the facial recognition workflow: the face is detected, aligned to a standard pose using facial landmarks, converted to a descriptor by a convolutional network and compared with stored templates. The design requires explicit consent before a face is captured.

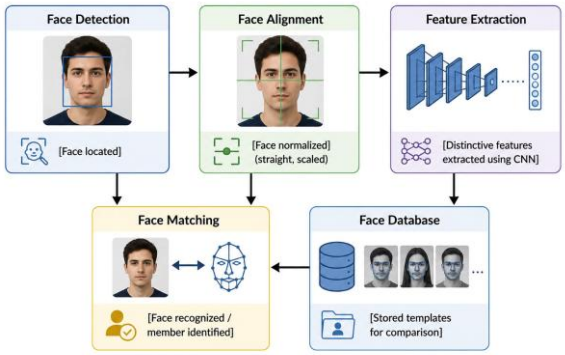


Figure 2.5: Facial recognition workflow for member authentication.

Figure 7. Facial recognition workflow: detection, alignment, feature extraction, matching against stored templates.

Figure 8 shows the recommendation flow. When a member borrows a book, the event is recorded; the engine reads the book's metadata, applies content based filtering (similar authors and categories), collaborative filtering (books borrowed by members with similar histories) and a popularity score, combines them into a recommendation score, removes books already read and displays the top suggestions with a label explaining each one.

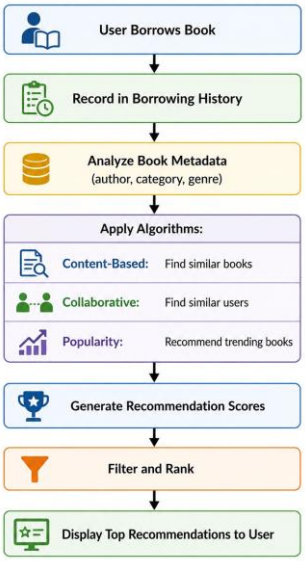


Figure 2.8: Book recommendation process flow from borrowing to suggestions.

Figure 8. Book recommendation process from borrowing event to displayed suggestions.

Figure 9 shows the voice search flow: the user speaks a query such as "find books by Chinua Achebe", the microphone captures audio, the Web Speech API performs acoustic modelling, language modelling and decoding, and the resulting text is run against the book database and the results are displayed.

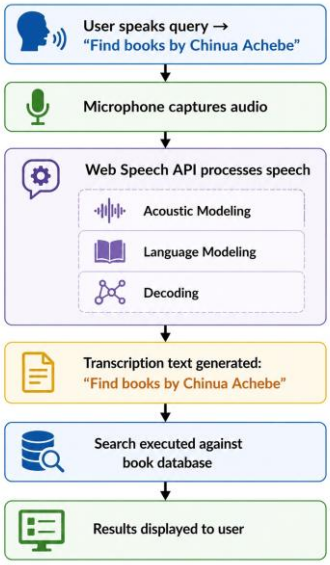


Figure 2.9: Voice search implementation using the Web Speech API.

Figure 9. Voice search flow using the Web Speech API.

3.6 Security design

Security controls were specified for each threat. Unauthorised access is countered by face authentication and role based access for librarians and members; injection and cross site scripting by input validation and HTML escaping; data loss by an export function for backups; and privacy violation by processing face data only on the client. Data interception is countered by serving the system over HTTPS. The design notes that LocalStorage can be encrypted with the Web Crypto API, which the current version does not yet do. Table 5 lists the security controls specified for each threat and how they were implemented.

Table 5. Security threats and controls

Threat	Control	Implementation
Unauthorised access	facial recognition, roles	descriptor match, role checks
Data interception	HTTPS	TLS on the static host
Injection attacks	input validation	sanitising of inputs
Cross site scripting	output escaping	HTML escaping function
Data loss	regular backup	export function
Privacy violation	client side processing only	no data sent to a server

The controls in Table 5 protect against outside attackers but not against someone with access to the desk computer, who can read LocalStorage directly. This matters most for face descriptors, which Section 7 discusses.

4 Implementation and Testing

4.1 Core modules

The system was implemented as a set of HTML pages sharing JavaScript modules. Figure 10 shows the dashboard. Summary tiles report total books, borrowed books and members, with change indicators, and a panel shows recently added books. The sidebar groups pages into main, management and reports sections, and a theme switch toggles light and dark modes.

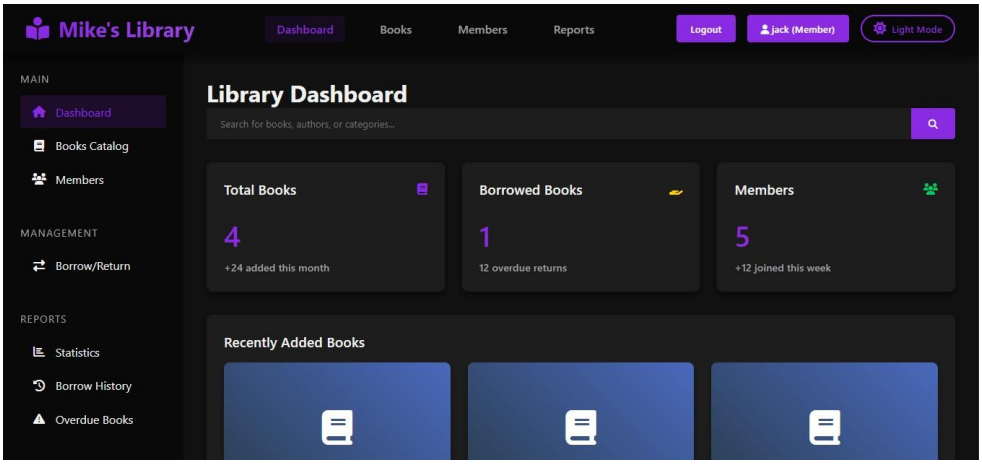


Figure 10. Library dashboard with summary tiles for books, borrowed books and members, and recently added books.

The dashboard in Figure 10 gives a librarian the state of the collection at a glance. The figures in the screenshot (four books, one borrowed, five members) come from the test collection used during development, which is far smaller than a working library; Section 5 therefore evaluates the AI modules on larger data.

Figure 11 shows the books catalogue. Each book is shown as a card with its cover area, title, author, year and availability, and buttons allow filtering and adding books. Every create, update or delete operation rewrites the corresponding array in LocalStorage as a JSON string, and the cards are regenerated so that event handlers stay attached.

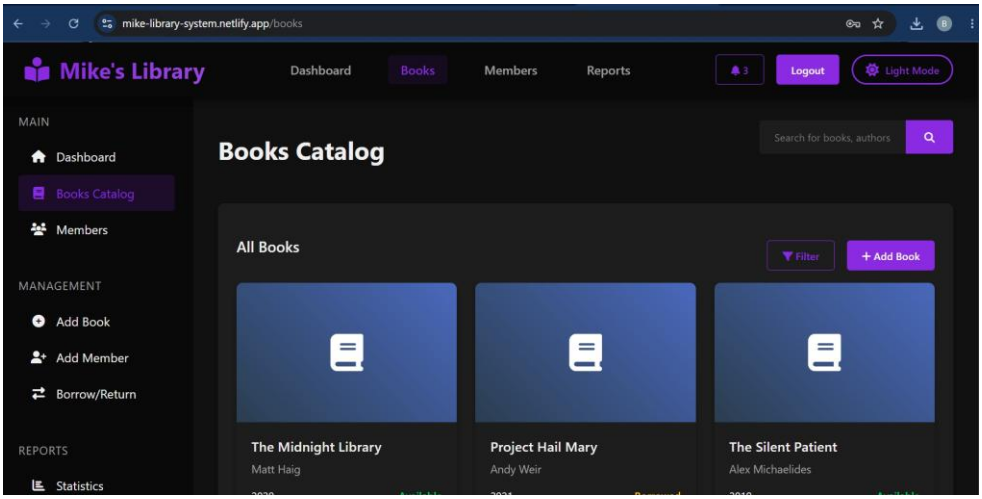


Figure 11. Books catalogue showing book cards with title, author and availability, and the add book function.

Figure 12 shows the borrow dialog. The librarian selects the member from a list, or uses face login to identify the member, and confirms. The system marks the book as borrowed, records the member's name, sets the due date fourteen days ahead and appends a record to the borrowing history. A return reverses the status and logs the return.

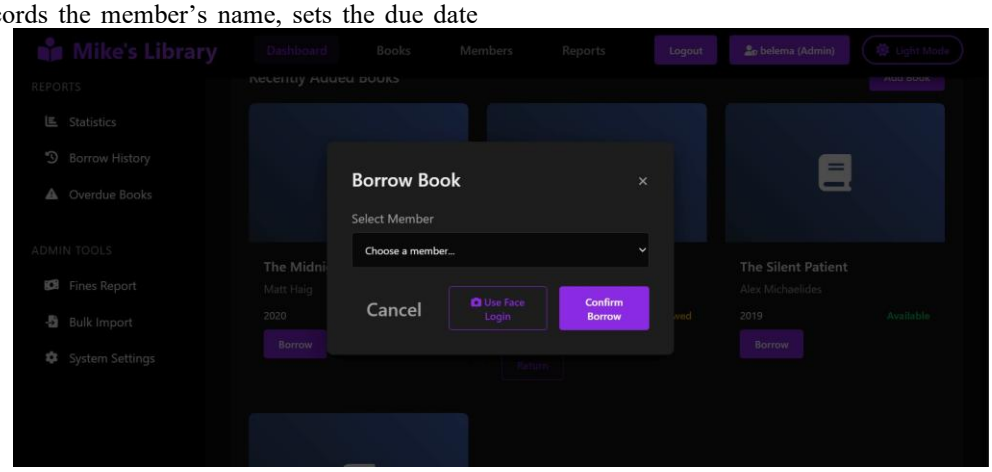


Figure 12. Borrow dialog with member selection, face login option and confirmation.

The dialog in Figure 12 is the point where the AI features meet the core workflow: the member can be chosen from the list or identified by face, and the book can be chosen from the catalogue or by scanning its barcode. Keeping the manual path beside the automatic one means a failed scan or an unrecognised face slows the desk down but never stops it.

Figure 13 shows the recommendation panel for a selected member. The engine scores every unread book by three components: a content score for books whose author or category the member has borrowed before, a collaborative score for books borrowed by members who share books with this member, and a popularity score from the total number of borrows. It shows the four highest scoring books, each with a label such as *Recommended for you*.

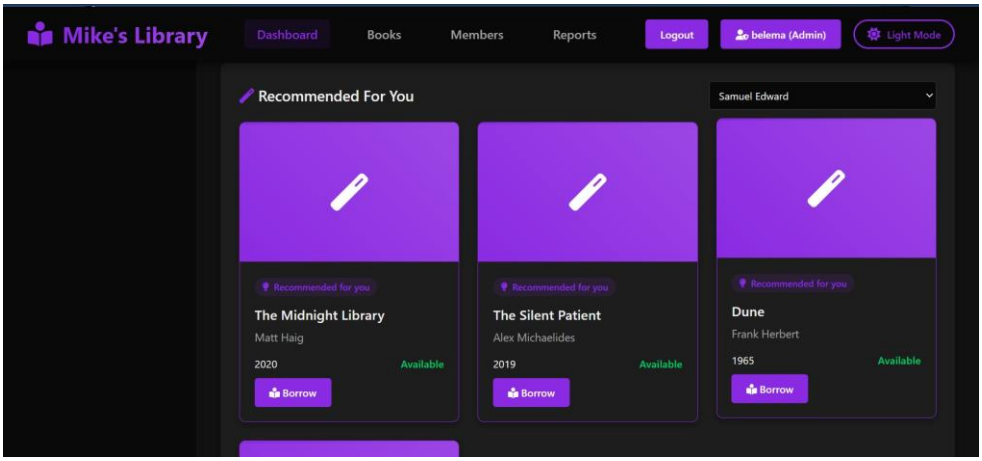


Figure 13. Recommendation panel showing personalised suggestions for a selected member, each with a borrow button.

The panel in Figure 13 shows four suggestions with a borrow button on each, so a member can act on a suggestion at once. With only a handful of books in the test collection, every member receives nearly the same suggestions, which is why Study B measures the engine on data with 2,000 titles and over 50,000 readers.

Figure 14 shows the category heat map, drawn with Chart.js as a horizontal bar chart. The librarian can switch between borrowing frequency, number of books held and a popularity ratio. Bars are coloured red for high activity, yellow for medium and green for low, and the colours adapt to the light and dark themes. In the screenshot, fiction and mystery are the most borrowed categories and science fiction follows.

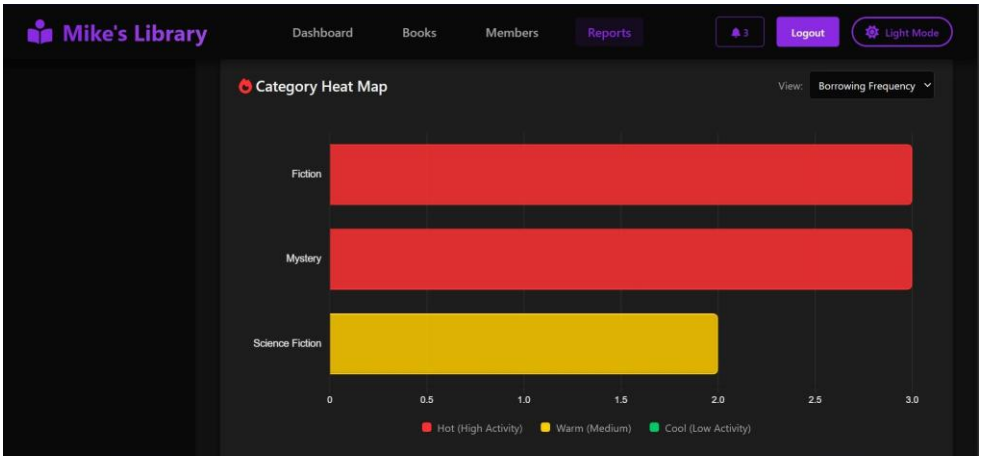


Figure 14. Category heat map showing borrowing frequency by category, coloured by activity level.

The heat map in Figure 14 turns the borrowing history into a picture of demand by subject. With real circulation data it would show librarians which shelves need more copies and which subjects are seldom used. The predictive analytics module uses the same counts: it computes the average borrowing rate per category and flags categories above the average as high demand for the next period, a simple rule that can be replaced by moving average or regression forecasts [13] once a library has several terms of history.

Figure 15 shows face registration. The FaceAPI.js library loads three pretrained models from a content delivery network: a face detector, a 68 point landmark model and the recognition network. The live camera feed is analysed, a single face is detected and outlined, and when the librarian presses Capture and Save, a 128 dimensional descriptor is stored in the member's record. At login a new descriptor is computed and compared with every stored descriptor by Euclidean distance; the member whose descriptor is closest is authenticated if the distance is below 0.55. In the published figures, the face and the member's name have been obscured to protect privacy.

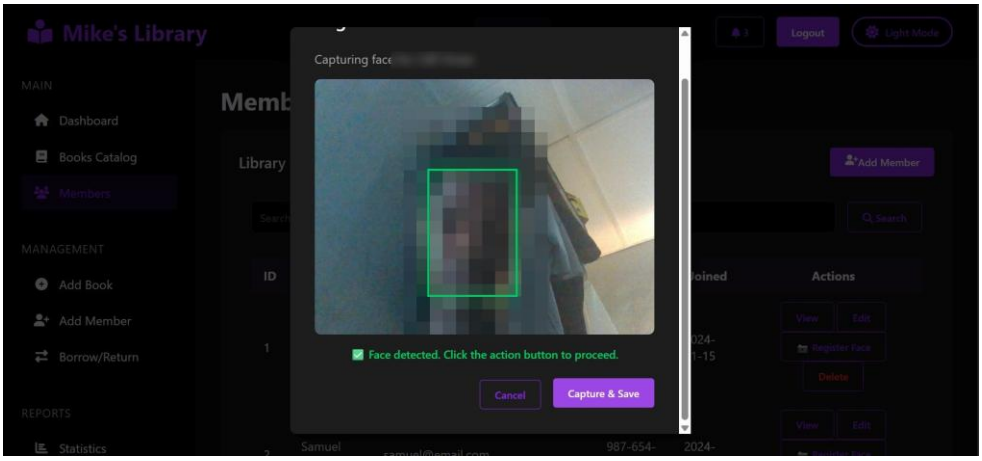


Figure 15. Face registration: the detected face is outlined and the descriptor is saved to the member's record (face and name obscured for privacy).

Figure 15 shows the detector working on a webcam image in indoor light: the face is found and boxed, and the message confirms that capture can proceed. The 0.55 threshold is slightly stricter than the 0.6 suggested for this descriptor family, which lowers the chance of accepting the wrong person at the cost of occasionally asking a genuine member to try again.

Figure 16 shows the barcode scanner. The dialog asks the user to point the camera at the barcode on the back of the book and draws a target frame on the live video. The Html5Qrcode library decodes the EAN 13 barcode, the ISBN is matched against the catalogue and, if it is found, the borrow dialog opens with the book selected. A manual ISBN field allows checkout when the camera cannot read the code.

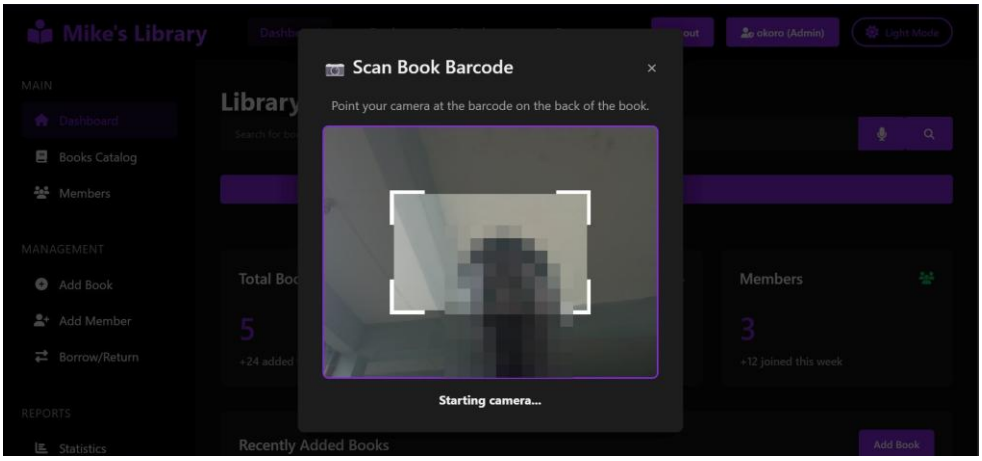


Figure 16. Barcode scanning dialog with live camera view and target frame (face in the camera view obscured).

The screenshot in Figure 16 was taken while the camera was starting, with no book in view. It shows the scanning frame within which the barcode must be placed. How reliably the decoder reads a barcode inside that frame depends on light, focus, distance and angle, which Study A measures.

4.2 Unit testing

Table 6 lists the ten unit tests. Each core function was tested with a defined input and expected result, and all ten passed.

Table 6. Unit test cases and results

Test	Description	Expected result	Actual result	Status
UT 01	Add a new book	Book appears in catalogue	Book appears correctly	Passed
UT 02	Add a new member	Member appears in table	Member appears correctly	Passed
UT 03	Borrow a book	Status changes to borrowed	Status updated and due date set	Passed
UT 04	Return a book	Status changes to available	Status updated and history logged	Passed
UT 05	Voice search	Search runs on spoken query	Results displayed correctly	Passed
UT 06	Recommendation engine	Suggestions based on history	Relevant books shown with reasons	Passed
UT 07	Heat map toggle	Chart updates with selection	Chart changes smoothly	Passed
UT 08	Face registration	Facial descriptor saved	Member profile updated	Passed
UT 09	Face login	User authenticated by webcam	Borrowing processed	Passed
UT 10	Barcode scanner	ISBN read, borrow dialog opens	Dialog opened correctly	Passed

The tests in Table 6 confirm that each feature works as designed on the development machine with the test collection. They do not measure how often the AI features succeed: a single successful face login or barcode scan shows that the code path works, not how it behaves in poor light or with hundreds of members. That is the purpose of Section 5.

4.3 Integration and performance testing

Integration tests confirmed that the modules work together, for example that a face login correctly fills the borrow dialog and that role based access hides administrative pages from members. Performance was measured on the development machine. The application loaded in 1.2 seconds on average, voice search returned results in 1.5 seconds and the face models took about 3 seconds to load the first time, after which the browser cache made them available at once. All are within the 3 second requirement.

4.4 Problems encountered

Four problems arose during implementation. The Web Speech API ended recognition early with a no speech error because of its default microphone timeout; an automatic retry fixed this. Virtual audio devices on the test machine captured the microphone input; users were told to select their physical microphone in the browser settings. The face models are several megabytes and slow to download; loading them from a reliable content delivery network and caching them in the browser reduced the delay. Barcodes were hard to read in low light; a manual ISBN field was added as a fallback. The last problem motivated Study A.

5 Evaluation Method

5.1 Study A: reliability of barcode checkout

The aim was to find the camera conditions under which barcode checkout fails, and to test a remedy. We generated 150 random valid ISBN 13 numbers (prefix 978 with a correct check digit) and rendered each as an EAN 13 barcode at 300 dots per inch with a module width of 0.4 mm, giving about 4.7 pixels per bar module, which corresponds to a book held 20 to 30 cm from a 720p webcam. Each barcode was placed on a mid grey background to stand for a book cover. We then degraded the images one factor at a time: illumination (pixel values multiplied by a factor from 1.0 down to 0.05, with mild sensor noise), defocus blur (Gaussian, sigma 0 to 6 pixels), sensor noise (standard deviation 0 to 90 grey levels), in plane rotation (0 to 90 degrees), resolution (images scaled from 1.0 down to 0.25) and perspective tilt (up to 30 percent of the width). Figure 17 shows examples. We also defined six combined scenarios typical of a library, from a well lit desk to a dim aisle with a low cost webcam.



Figure 17. Examples of the simulated capture conditions applied to the same ISBN barcode.

Figure 17 shows that the conditions range from easy to severe. At 20 percent light the bars are barely visible to the eye, and at a scale of 0.4 each bar module is less than two pixels wide.

Each image was decoded by the ZXing C++ decoder [32], the native implementation of the algorithm that Html5Qrcode runs in the browser. We compared two pipelines: the *baseline* passes the image to the decoder as captured, as the system does now; the *enhanced* pipeline, used only when the baseline fails, applies a 3 by 3 median filter and a contrast stretch that maps the darkest and brightest 1 percent of pixels to black and white, and then retries at rotations of 0, plus or minus 30 and plus or minus 60 degrees. A decode counts as correct only if the decoded digits equal the true ISBN; a different number counts as a misread. We report the share decoded correctly, the misread rate and the median decoding time per image.

5.2 Study B: offline evaluation of the recommender

The test collection in the system is too small to evaluate recommendations, and no Nigerian library circulation data were available to us. We therefore used goodbooks 10k [31], a public dataset of about six million ratings of 10,000 books by 53,424 readers, with author names and the shelf tags readers gave each book. We treated every rated book as a book the reader had borrowed. To match the size of a university library, we kept the 2,000 most read titles and readers with at least 10 of them, leaving 4,032,581 records from 53,120 readers (median 75 books per reader).

We drew 4,000 test readers at random and hid 20 percent of each one’s books (62,179 hidden records); all other readers’ records were used in full. Four methods, matching the components of the system’s engine, ranked all unread books for each test reader:

1. *Popularity*: the number of readers of each book, as used for new members.

2. *Content based*: each book was described by its first author and its 30 most frequent shelf tags (excluding tags such as to read or owned), weighted by the logarithm of the tag count and normalised; a reader’s score for a book is the sum of its cosine similarity to the books the reader has read.

3. *Collaborative*: item to item cosine similarity of co reading [24], summed over the books the reader has read.

4. *Hybrid*: the sum of the three scores after scaling each to the range 0 to 1 for every reader, first with equal weights, as in the system, and then with weights chosen on half of the test readers from a grid of values and reported on the other half.

The system shows four suggestions, so each method's top four books were compared with the hidden books. We report precision at 4 (the share of the four suggestions that the reader actually read), recall at 4, hit rate at 4 (the share of readers with at least one correct suggestion), normalised discounted cumulative gain at 4 [16], and catalogue coverage, the share of the 2,000 titles that appear in anyone's top four. All figures are computed on the 2,000 readers not used for tuning. To study new members, we also gave each method only 1, 3, 5 or 10 known books per reader.

Both studies were run in Python with a fixed random seed of 42 on a two core machine, and together take about six minutes.

6 Results

6.1 Barcode checkout

Table 7 summarises the decoding rates and Figure 18 plots them for every condition. In normal light the baseline decoder read every barcode, and it kept doing so with perspective tilt up to 30 percent of the width, rotation up to 20 degrees, noise up to a standard deviation of 50 grey levels and scaling down to 0.5. It failed in four situations: at 12 percent of normal light or less, at a blur of 2 pixels or more, at rotations between 30 and 60 degrees, and when the image was scaled to 0.3 or less, where it read only 18.7 percent.

Table 7. Share of 150 ISBN barcodes decoded correctly (percent), baseline decoder and enhanced pipeline

Condition	Baseline	Enhanced
Illumination 1.0 of normal	100.0	100.0
Illumination 0.3 of normal	94.7	99.3
Illumination 0.2 of normal	100.0	100.0
Illumination 0.12 of normal	0.0	100.0
Illumination 0.08 of normal	0.0	99.3
Illumination 0.05 of normal	0.0	88.0
Blur sigma 1	100.0	100.0
Blur sigma 2	0.0	98.7
Blur sigma 3	0.0	0.0
Noise SD 50	100.0	100.0
Noise SD 70	92.0	100.0
Noise SD 90	89.3	98.7
Rotation 20 degrees	100.0	100.0
Rotation 30 degrees	0.0	98.7
Rotation 45 degrees	0.0	100.0
Rotation 60 degrees	0.0	100.0
Scale 0.5	100.0	100.0
Scale 0.4	96.0	96.0
Scale 0.3	18.7	18.7
Scale 0.25	0.0	0.0
Tilt 0.3 of width	100.0	100.0
Scenario: well lit desk	100.0	100.0
Scenario: typical reading room	100.0	100.0
Scenario: dim stack aisle	0.0	98.7
Scenario: dim aisle, shaking hand	0.0	0.0
Scenario: low cost webcam, far	0.0	0.0
Scenario: very dark, low cost webcam	0.0	0.0

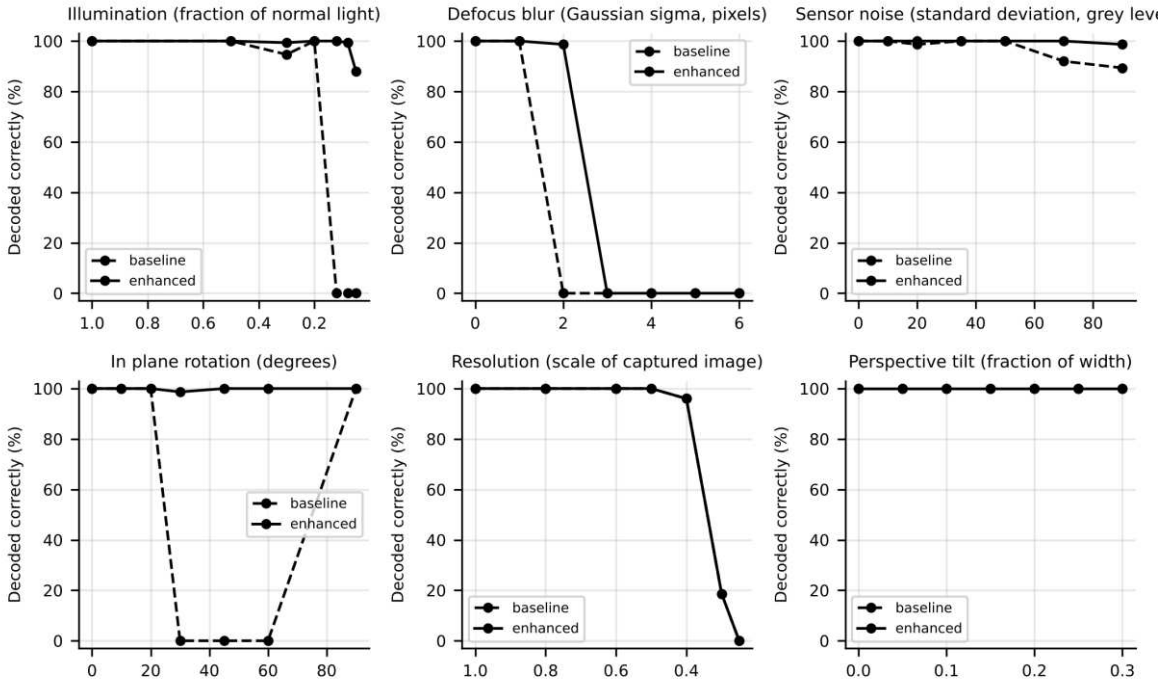


Figure 18. Share of 150 ISBN barcodes decoded correctly under each simulated condition, for the baseline decoder (dashed) and the enhanced pipeline (solid).

The enhanced pipeline removed three of these failures. Contrast stretching recovered dim images: decoding rose from 0 to 100 percent at 12 percent light, to 99.3 percent at 8 percent and to 88.0 percent at 5 percent. Median filtering recovered mild blur: at sigma 2, decoding rose from 0 to 98.7 percent. The rotated retries recovered rotation: from 0 to between 98.7 and 100 percent at 30, 45 and 60 degrees. Heavy blur (sigma 3 or more) and low resolution (scale 0.3 or less) remained unreadable, because the bars themselves are lost and no processing of the captured image can restore them.

Figure 18 shows how sharp the failures are. Each curve stays near 100 percent and then falls to zero within one step, so a reader either works reliably or not at all, and small changes in the desk lighting or the distance at which members hold books can decide which. Misreads were rare in both pipelines: at most 2.0 percent of codes at a scale of 0.4 and 1.3 percent at 30 degrees with enhancement, and none in normal conditions. Because the system only accepts an ISBN that exists in its catalogue, a misread would almost always be rejected rather than lend the wrong book.

In the combined scenarios (Table 7, lower part), both pipelines read every code at a well lit desk and in a typical reading room. In a dim stack aisle, the baseline read none and the enhanced pipeline 98.7 percent. A dim aisle combined with a shaking hand (blur 3), a distant low cost webcam and a very dark scene with a low cost webcam defeated both pipelines. The enhancement costs time only when the first attempt fails: the median decoding time stayed at about 3 milliseconds in good conditions and rose to about 98 milliseconds in the dim aisle, still far below what a user would notice.

6.2 Recommendation

Table 8 gives the results for the 2,000 evaluation readers. Popularity, which suggests the same four books to everyone, was correct for 12.8 percent of suggestions, and 36.0 percent of readers had at least one of the four among their hidden books. Content based filtering by author and tags did only slightly better, with a precision of 15.1 percent. Item to item collaborative filtering more than doubled precision, to 33.6 percent, and gave at least one correct suggestion to 70.1 percent of readers.

Table 8. Recommendation accuracy on goodbooks 10k (2,000 titles; 2,000 evaluation readers, 20 percent of each reader's books hidden). Four suggestions per reader; standard error of precision in brackets.

Method	Precision at 4 (%)	Recall at 4 (%)	Hit rate at 4 (%)	NDCG at 4	Catalogue coverage (%)
Popularity	12.8 (0.4)	3.2	36.0	0.134	1.1
Content based (author and tags)	15.1 (0.5)	4.6	42.6	0.157	36.4
Collaborative (item to item)	33.6 (0.7)	9.5	70.0	0.361	20.3
Hybrid, equal weights	26.6 (0.6)	7.4	62.4	0.285	11.6
Hybrid, tuned weights	33.8 (0.7)	9.6	70.7	0.363	21.9

The hybrid with equal weights, the combination first implemented in the system, reached only 26.6 percent, worse than collaborative filtering alone. Adding the popularity score pulls every reader's list toward the same bestsellers, and adding the weaker content score dilutes the stronger collaborative signal. When the weights were chosen on separate readers, the best combination gave the collaborative score a weight of 1, the content score 0.25 and popularity 0. This tuned hybrid reached 33.8 percent precision, 70.7 percent hit rate and the highest NDCG (0.363), a small but consistent gain over collaborative filtering alone. It also recommended 21.9 percent of the catalogue, compared with 1.1 percent for popularity; content based filtering spread its suggestions most widely (36.5 percent) but was seldom right.

Figure 19 shows what happens for members with short histories. With one known book, collaborative filtering reached 17.2 percent and the tuned hybrid 16.4 percent, both above popularity (12.8 percent), while content based filtering fell below popularity (10.6 percent). With three known books the collaborative and hybrid methods already reached about 26 percent, and with ten about 32 percent, close to their full history results.

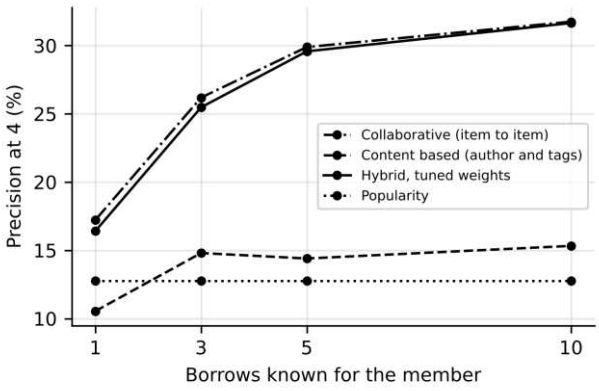


Figure 19. Precision at 4 for members whose recommendations are based on only 1, 3, 5 or 10 known borrows.

Figure 19 shows that popularity is not needed as a fallback once a member has borrowed even one book in a library with an active community of borrowers: a single borrowed book is enough for co reading patterns to beat the bestseller list. Popularity remains the only option for a member with no history at all.

7 Discussion

7.1 What the results mean for the system

The two studies show that the AI parts of a library system can be judged in the same way as any other component, by how often they succeed under the conditions they will meet. For barcode checkout, the decoder is reliable in ordinary reading room light and at ordinary angles, so it

will speed up the desk in most cases. It fails abruptly in dim light, with mild blur and with books held at an angle, which matches the low light problem met during development. A few lines of image processing before a second attempt restore it in most of these cases at negligible cost, and we recommend adding them to the scanner. Blur from a moving hand and very low resolution cannot be fixed afterwards; for these, the interface should ask the user to hold the book still and closer, and the desk should have a fixed light and a 1080p camera, which costs little compared with a laser scanner.

For recommendations, the evidence favours collaborative filtering, with a small weight on content and no weight on popularity once a member has any history. This changes the first implementation, which weighted the three parts equally and lost a fifth of its precision as a result. The tuned weights should be rechecked on the library's own data once it has a term or more of circulation records, because reading patterns in a Nigerian academic library, where course lists drive much borrowing, may differ from those of the goodbooks readers. A useful property of item to item filtering is that each suggestion has a simple explanation (members who borrowed this also borrowed that), which supports the labels the system already shows.

7.2 Face recognition, privacy and security

Face login was the feature members noticed most, but it carries the greatest risks. Three need attention before use with real members. First, the system has no liveness check, so a photograph of a member held up to the camera may be accepted; presentation attack detection, for example by asking the member to blink or turn the head, should be added [21]. Second, accuracy of face recognition models varies across demographic groups, and published evaluations found higher error rates for darker skinned faces [2, 9]; the 0.55 threshold should be validated with consenting members of the library's own population, measuring false acceptance and false rejection, before it is relied upon. Third, a face descriptor is biometric data, which the Nigeria Data Protection Act 2023 [7] treats as sensitive personal data requiring explicit consent and appropriate security. Storing descriptors unencrypted in LocalStorage on a shared desk computer does not meet that standard. Descriptors should be encrypted with the Web Crypto API, deletable on request, and never exported in backups without encryption. Face login should remain optional, with the member list and card number available to anyone who declines.

7.3 From one browser to a library service

The client only design kept the system free and private, but it means each desk computer holds its own data. A library with several desks, or members who want to see their loans on their phones, needs a shared database. A small server or a hosted database with access rules would allow this while keeping face matching on the client, so that descriptors need not be sent over the network. The same change would remove the LocalStorage size limit and allow the recommendation and forecasting modules to learn from the whole library rather than one desk.

7.4 Limitations

Study A used synthetic barcodes and simulated degradations rather than photographs of real books; real covers add glare, curvature and printed patterns near the code, so the absolute rates are optimistic, although the relative effects of light, blur and rotation, and of the enhancement, should carry over. The ZXing C++ decoder implements the same method as the JavaScript library in the browser but may differ in detail. Study B used ratings from an online reading community as a stand in for borrowing, which differ from library loans in motivation and scale; it measures how well each method predicts books a reader will read, not whether members will find the suggestions useful. We did not measure face recognition accuracy, voice search accuracy or forecast error, which require data from the library's own members and circulation. The functional tests were run on a small test collection by the developers.

7.5 Recommendations

We recommend that libraries adopting the system (1) add the enhancement and retry step to the barcode scanner and install a fixed light and a 1080p camera at the desk; (2) weight collaborative filtering most heavily in the recommender, keep a small content weight and use popularity only for members with no history; (3) add liveness detection, encrypt stored face descriptors, obtain explicit consent and validate the face threshold on the local population; (4) move storage to a shared database with access control; and (5) collect circulation data for at least a term before relying on the demand forecasts.

8 Conclusion

This paper presented an AI powered smart library management system that runs entirely in the browser and combines barcode based computer vision checkout, face login, voice search, hybrid book recommendation, a category heat map and demand forecasting at no licence cost. All ten unit tests passed, pages loaded in 1.2 seconds and voice queries returned in 1.5 seconds. A controlled study of 150 ISBN barcodes showed that camera checkout is reliable in normal conditions but fails abruptly in dim light, with mild blur or at angles of 30 to 60 degrees, and that median filtering, contrast stretching and rotated retries raise decoding from 0 to about 99 percent in these cases. An offline study on 4.0 million reading records showed that collaborative filtering more than doubles the precision of popularity and content matching, that equal weighting of the hybrid loses accuracy, and that tuned weights reach a precision of 33.8 percent and a hit rate of 70.7 percent with four suggestions. Future work will move the system to a shared database, add liveness detection and encrypted biometric storage, and evaluate face recognition, voice search and forecasting with members and circulation data from the University of Port Harcourt library.

Data and Code Availability

The goodbooks 10k dataset is publicly available from its author. The scripts that generate every result in Section 6, with a fixed random seed, and the system source code are available from the authors.

References

[1] Asemi, Asefeh, Ko, Andrea, and Nowkarizi, Mohsen. 2021. Intelligent Libraries: A Review on Expert Systems, Artificial Intelligence, and Robot. *Library Hi Tech* 39, 2 (2021), 412–434. doi:10.1108/LHT-02-2020-0038

[2] Buolamwini, Joy and Gebru, Timnit. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, 77–91.

[3] Burke, Robin. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User Adapted Interaction* 12, 4 (2002), 331–370. doi:10.1023/A:1021240730564

[4] Cox, Andrew M., Pinfield, Stephen, and Rutter, Sophie. 2019. The Intelligent Library: Thought Leaders' Views on the Likely Impact of Artificial Intelligence on Academic Libraries. *Library Hi Tech* 37, 3 (2019), 418–435. doi:10.1108/LHT-08-2018-0105

[5] Cremonesi, Paolo, Koren, Yehuda, and Turrin, Roberto. 2010. Performance of Recommender Algorithms on Top N Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys)*, 39–46. doi:10.1145/1864708.1864721

[6] Davis, Fred D.. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13, 3 (1989), 319–340. doi:10.2307/249008

[7] Federal Republic of Nigeria. 2023. Nigeria Data Protection Act, 2023. Official Gazette, Abuja.

[8] Gallo, Orazio and Manduchi, Roberto. 2011. Reading 1D Barcodes with Mobile Phones Using Deformable Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 9 (2011), 1834–1843. doi:10.1109/TPAMI.2010.229

[9] Grother, Patrick, Ngan, Mei, and Hanaoka, Kayee. 2019. Face Recognition Vendor Test (FRVT) Part 3: Demographic Effects. doi:10.6028/NIST.IR.8280

[10] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[11] Howard, Andrew G., Zhu, Menglong, Chen, Bo, Kalenichenko, Dmitry, Wang, Weijun, Weyand, Tobias, Andreetto, Marco, and Adam, Hartwig. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

[12] Huang, Gary B., Ramesh, Manu, Berg, Tamara, and Learned-Miller, Erik. 2007. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.

[13] Hyndman, Rob J. and Athanasopoulos, George. 2021. Forecasting: Principles and Practice. OTexts, Melbourne.

[14] Ingwersen, Peter and J\"arvelin, Kalervo. 2005. The Turn: Integration of Information Seeking and Retrieval in Context. Springer, Dordrecht.

[15] International Organization for Standardization. 2009. ISO/IEC 15420:2009 Information Technology, Automatic Identification and Data Capture Techniques, EAN/UPC Bar Code Symbology Specification. ISO, Geneva.

[16] J\"arvelin, Kalervo and Kek\"al\"ainen, Jaana. 2002. Cumulated Gain Based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446. doi:10.1145/582415.582418

[17] King, Davis E.. 2009. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research* 10 (2009), 1755–1758.

[18] Linden, Greg, Smith, Brent, and York, Jeremy. 2003. Amazon.com Recommendations: Item to Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. doi:10.1109/MIC.2003.1167344

[19] Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott, Fu, Cheng-Yang, and Berg, Alexander C.. 2016. SSD: Single Shot MultiBox Detector. In *Computer Vision, ECCV 2016*. Springer, 21–37.

[20] Norman, Donald A.. 2013. The Design of Everyday Things. Basic Books, New York.

[21] Ramachandra, Raghavendra and Busch, Christoph. 2017. Presentation Attack Detection Methods for Face Recognition Systems: A Comprehensive Survey. *ACM Computing Surveys* 50, 1 (2017), 8:1–8:37. doi:10.1145/3038924

[22] Redmon, Joseph, Divvala, Santosh, Girshick, Ross, and Farhadi, Ali. 2016. You Only Look Once: Unified, Real Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

[23] . 2015. Recommender Systems Handbook. Springer, New York.

[24] Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. 2001. Item Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, 285–295. doi:10.1145/371920.372071

[25] Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823.

[26] Smilkov, Daniel, Thorat, Nikhil, Assogba, Yannick, and others. 2019. TensorFlow.js: Machine Learning for the Web and Beyond. Proceedings of Machine Learning and Systems (SysML); arXiv:1901.05350.

[27] Smith, Ray. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 629–633. doi:10.1109/ICDAR.2007.4376991

[28] Venkatesh, Viswanath, Morris, Michael G., Davis, Gordon B., and Davis, Fred D.. 2003. User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly* 27, 3 (2003), 425–478. doi:10.2307/30036540

[29] W3C Audio Community Group. 2023. Web Speech API. W3C Community Group Draft Report.

[30] Wang, Mei and Deng, Weihong. 2021. Deep Face Recognition: A Survey. *Neurocomputing* 429 (2021), 215–244. doi:10.1016/j.neucom.2020.10.081

[31] . 2017. Goodbooks 10k: A New Dataset for Book Recommendations. \urlhttps://github.com/zygmuntz/goodbooks-10k.

[32] ZXing Project. 2024. ZXing (Zebra Crossing): Multi Format 1D/2D Barcode Image Processing Library. \urlhttps://github.com/zxing/zxing.