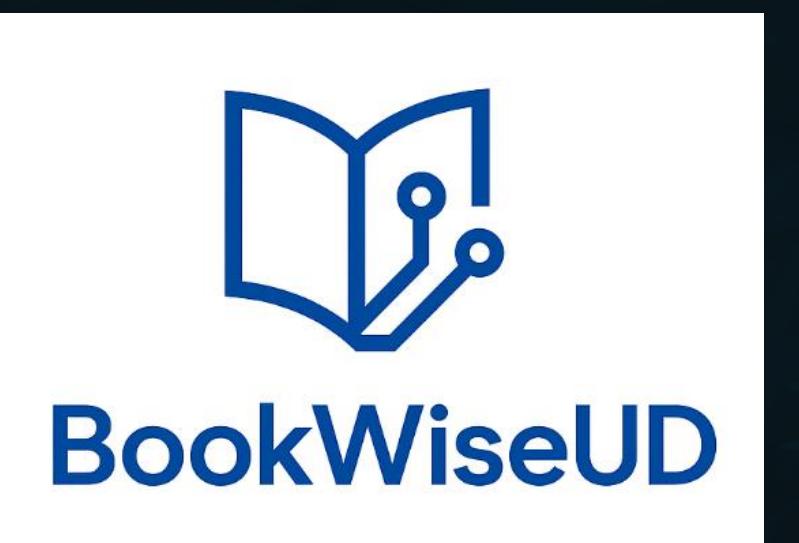


BookWiseUD: A Scalable Library Management System

Presented by:

Wilder Steven Hernandez Manosalva

Jhon Javier Castañeda Alvarado



Addressing Library Management Challenges

Manual Systems

Inefficiencies and data inconsistencies from paper-based or fragmented digital records.

Lack of Integration

No real-time inventory, automated loans, or secure user authentication.

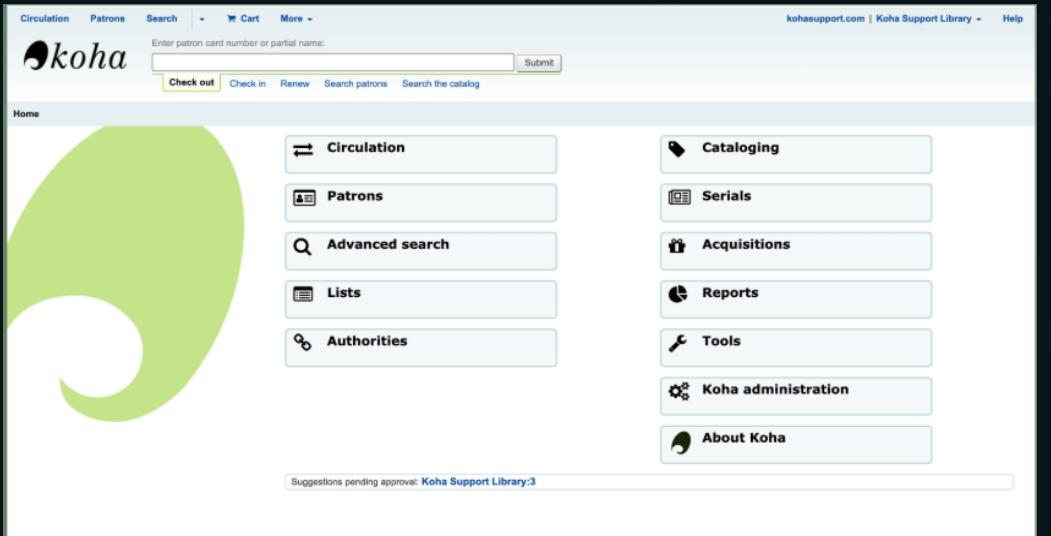
Poor User Experience

Patrons and administrators face bottlenecks and limited access to information.

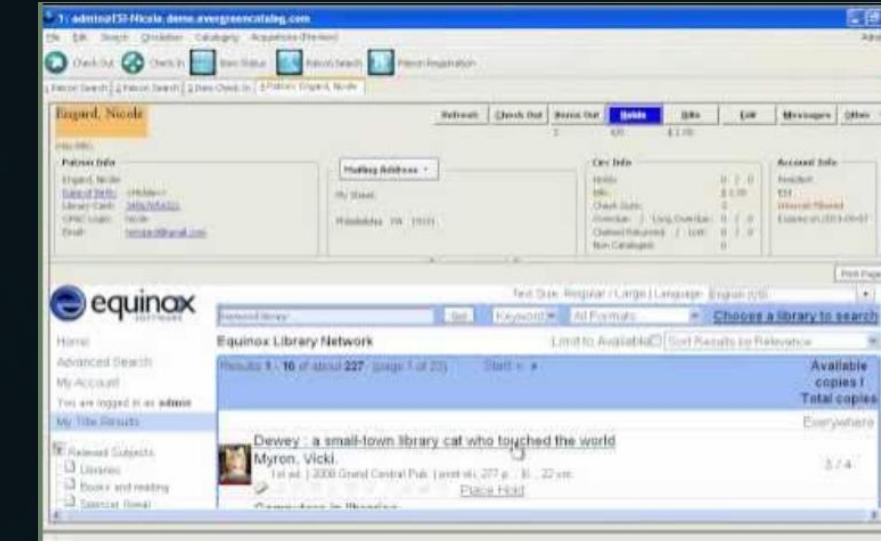


Base or references (inspiration)

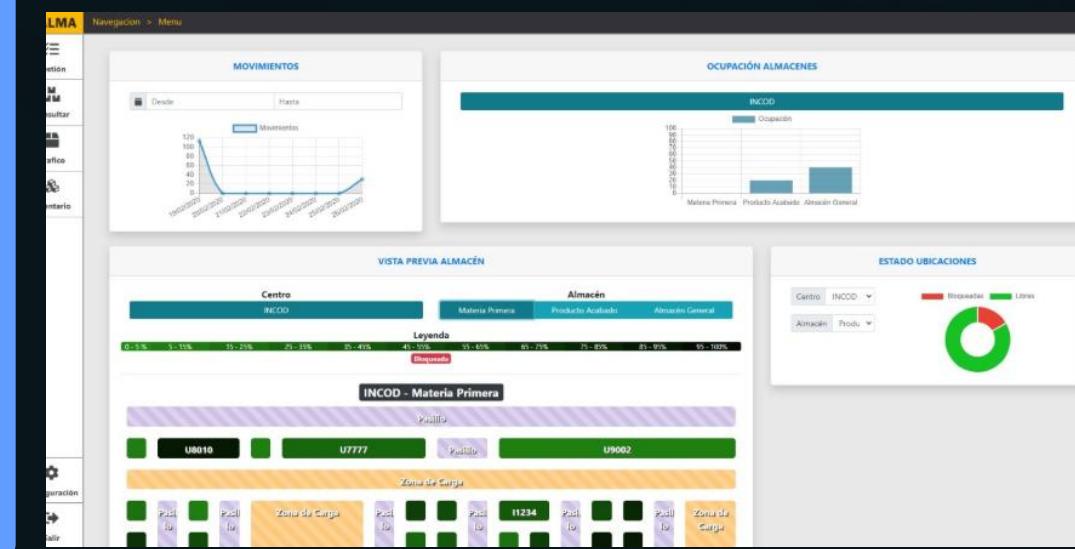
Koha Integrated Library System



Evergreen ILS



Sierra and alma software



Architectural Decisions: Dual-Service Approach

We chose a dual-service architecture over a monolith for specific advantages.

Security Isolation

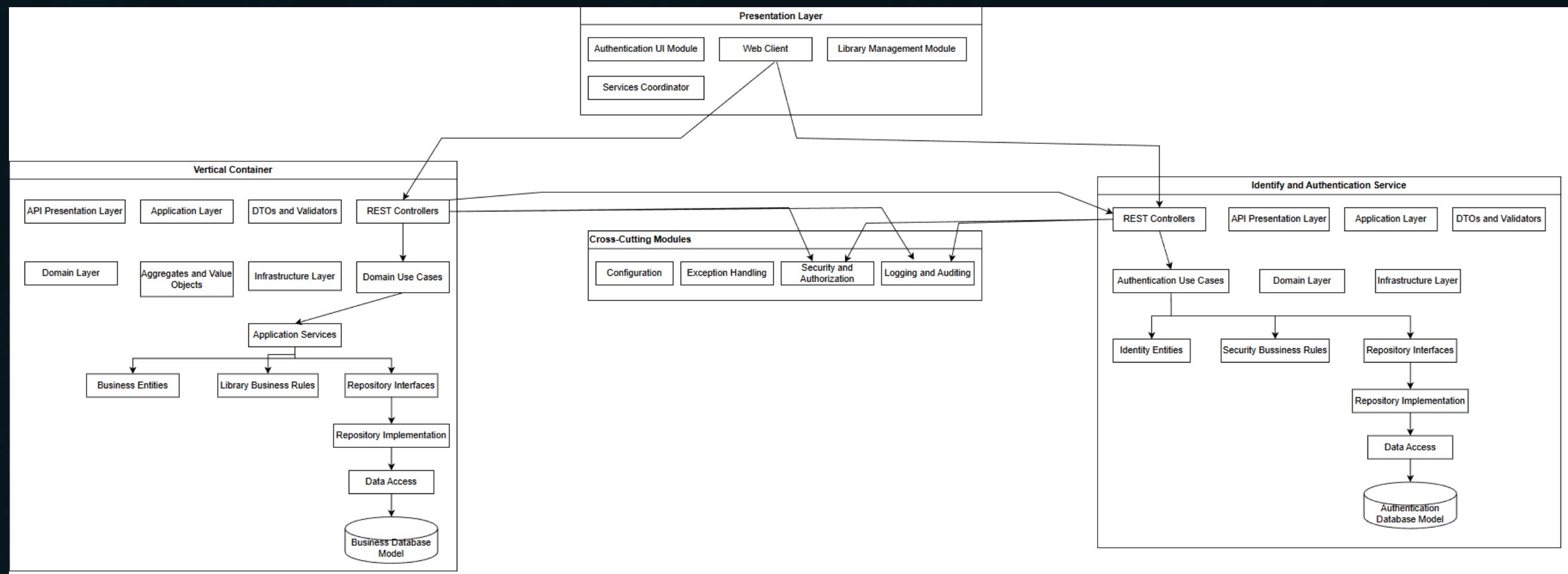
Credential storage separated from domain operations.

Technology Specialization

Spring Security for auth, FastAPI for performance.

Academic Objectives

Demonstrate distributed systems and microservices.



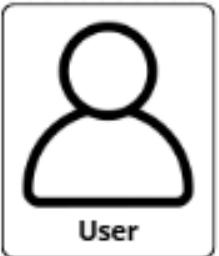
Users Stories

The distribution shows high-priority core features were implemented first. Librarian stories (57% of total) emphasize administrative control, while user stories focus on self-service access.

TABLE I
USER STORIES IMPLEMENTATION SUMMARY

ID	Story Name	Sprint	Priority	Points
Librarian Stories				
US01	Receive Return	1	High	3
US02	Lend Book	1	High	5
US03	Delete User	2	High	2
US04	Update User	2	High	3
US05	Register User	2	High	3
US06	Delete Book	3	High	2
US07	Update Book	3	High	3
US08	Register Book	3	High	3
User Stories				
US09	View Available Books	1	High	3
US10	Request Book Loan	2	High	5
US11	Return Borrowed Book	3	Medium	3
US12	View Loan History	3	Medium	2
US13	Log in to System	1	High	3
US14	Recover Account/Password	1	High	3

User Story Mapping



User

Book Catalog Access

As a user, I want to view the catalog of available books so that I can know which ones I can borrow.(US09)

Access catalog section.

Retrieve available books from database.

Display relevant info (title, author, category, status).

Implement filters by title, author, or category.

Handle "no available books" message.

Book Loan Request

As a user, I want to request the loan of an available book so that I can read it for a specific period.(US10)

Select a book from catalog.

Request loan through form or button.

Validate availability before confirmation.

Record loan details (date, user ID).

Display success or error message.

Book Return

As a user, I want to register the return of a borrowed book so that my loan history remains updated.(US11)

View current borrowed books.

Select a book to return.

Confirm return.

Update book status → Available.

Display confirmation or error message.

Loan History

As a user, I want to view my past loan history so that I can check the books I have borrowed.(US12)

Access loan history section.

Retrieve user loan data from database.

Display title, loan date, and return date.

Enable sorting by date or title.

Show message if no previous loans.

Authentication

As a user, I want to log in to the system using my credentials so that I can access features available for my role.(US13)

Access login form.

Enter email and password.

Validate credentials against database.

Manage session token creation and expiration.

Display success or error message.

Implement logout function.

Account Recovery

As a user, I want to recover my account when I forget my password so that I can regain access without losing my data.(US14)

Enter registered email.

System sends verification link or code.

Open recovery form via secure link.

Enter new password.

Validate and update credentials.

Show confirmation or error message.



Librarian

Book Return Management

As a librarian, I want to receive returned books from users, so that the book inventory is updated and the return process is recorded correctly.(US01)

Verify that the librarian session is active.

Retrieve the list of active loans from the database.

Display the list of active loans in the frontend.

Allow the librarian to select a specific loan record.

Display loan details (user, book, loan dates).

Provide a "Confirm Return" button.

On confirmation, update book status → Available.

Record return date and user ID in the database.

Show success or error message.

Validate all required fields before saving.

Book Loan Management

As a librarian, I want to lend books to users, so that the loan process is tracked and books are issued correctly.(US02)

Verify active librarian session.

Fetch all available books.

Display available books and user list.

Allow librarian to select book and user.

Record new loan (loan date, return date).

Update book status → Loaned.

Validate required information.

Notify success or failure.

Validate user existence before deletion.

User Management

As a librarian, I want to delete a user from the system, so that inactive or incorrect accounts are removed.(US03)

Verify active librarian session.

Search user by name or ID.

Display user information.

Display user details in editable form.

Allow editing of user fields (name, email, phone, etc.).

Save new user in database.

Remove user record from database.

Show confirmation or error message.

Show success or error message.

User Registration

As a librarian, I want to register new users, so that they can borrow books and access library services.(US05)

Verify active librarian session.

Fill out registration form.

Validate required fields (name, ID, email, etc.).

Display book information.

Display book by title or ID.

Display book data in database.

Display confirmation or failure message.

Save updated record.

Notify success or failure.

Book Inventory Management

As a librarian, I want to delete books from the system, so that obsolete or damaged books are removed from inventory.(US06)

Verify active librarian session.

Search book by title or ID.

Display book information.

Display editable book form.

Modify fields (title, author, category, year).

Prevent duplicate entries.

Show confirmation or failure message.

Save updated record.

Notify success or failure.

Book Registration

As a librarian, I want to register new books, so that they can be borrowed by users and tracked in the system.(US07)

Verify active librarian session.

Fill out book registration form.

Save book data in database.

Notify success or failure.

Validate input data.

Validate input data.

Save updated record.

Notify success or failure.

JIRA

The screenshot displays the Jira interface with two main panels. The left panel shows the 'Backlog' tab for the project 'project_seminar_engineering_software'. It lists various user stories (US) such as 'US13 Log in to the system', 'US14 Recover account or password', and several SCRM tasks related to authentication and system setup. The right panel shows the 'Tablero' (Kanban board) for the same project. The board has columns: TO DO, IN PROGRESS, IN REVIEW, TESTING, and DONE. The 'DONE' column contains 103 items, with the last few being: 'Validate that the book exists before deletion.', 'US01 Receive Return', 'Display book information.', 'Access catalog section.', 'Display user details in editable form.', 'Notify success or failure.', and 'SCRM-38'.

Left Panel (Backlog):

- Authentication and System Setu 20 oct - 27 oct (14 actividades)
- SCRUM-4 US13 Log in to the system
- SCRUM-5 US14 Recover account or password
- SCRUM-110 Access login form.
- SCRUM-114 Display success or error message.
- SCRUM-120 Enter new password.
- SCRUM-119 Open recovery form via secure link.
- SCRUM-112 Validate credentials against database.
- SCRUM-115 Implement logout function.
- SCRUM-117 Enter registered email.
- SCRUM-118 System sends verification link or code.
- SCRUM-121 Validate and update credentials.
- SCRUM-111 Enter email and password.
- SCRUM-113 Manage session token creation and expiration.
- SCRUM-122 Show confirmation or error message.

Right Panel (Tablero):

Column	Items
TO DO	+ Crear
IN PROGRESS	
IN REVIEW	
TESTING	
DONE	103

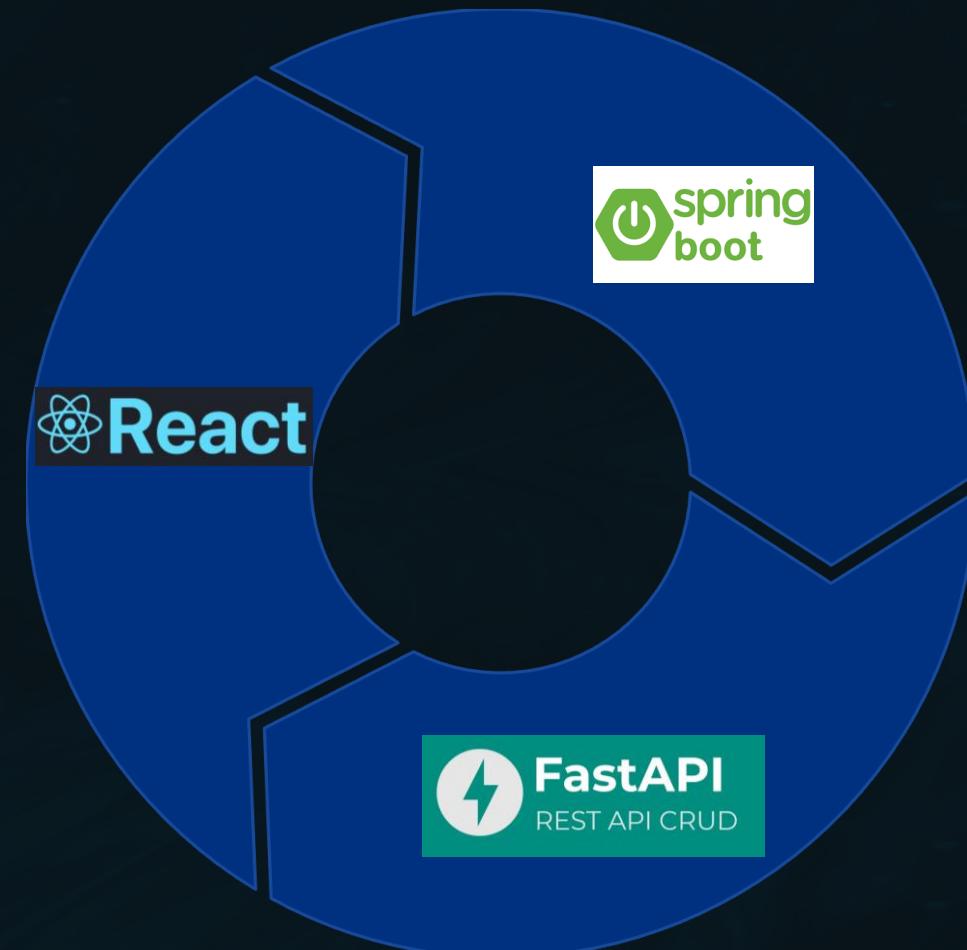
- Validate that the book exists before deletion. (Status: Done, assigned to SH)
- US01 Receive Return (Status: In Progress, assigned to JC)
- Display book information. (Status: In Progress, assigned to JC)
- Access catalog section. (Status: In Progress, assigned to JC)
- Display user details in editable form. (Status: In Progress, assigned to JC)
- Notify success or failure. (Status: In Progress, assigned to JC)
- SCRM-38 (Status: In Progress, assigned to JC)

BookWiseUD: Our Solution

A modular web-based system for Universidad Distrital Francisco José de Caldas.

React Frontend

Intuitive user interface for interaction.



Java Spring Boot

Authentication service with MySQL.

Python FastAPI

Operational service with PostgreSQL.

Key Functional Requirements

1 User Authentication

JWT token issuance and role-based access control.

2 Book & User Management

CRUD operations for librarians, user registration.

3 Loan Lifecycle

Creation, return, and history tracking.

4 Catalog & Search

Browsing and real-time availability for all authenticated users.



Validation & Testing Strategy

A multi-layered approach ensures reliability and performance.



Unit Testing

96% coverage for Python backend, 97% for Java auth service.



Acceptance Testing

100% pass rate for critical user journeys using Cucumber/Behave.



Stress Testing

4.84ms average response time under simulated load (JMeter).



CI/CD Automation

GitHub Actions pipeline for continuous integration and deployment.

Name	Stmts	Miss	Branch	BrPart	Cover	Missing
app__init__.py	0	0	0	0	100%	
app\auth_utils.py	58	3	18	1	95%	70, 99-100
app\crud__init__.py	0	0	0	0	100%	
app\crud\books.py	35	0	6	0	100%	
app\crud\category.py	11	5	0	0	55%	11-18
app\crud\loans.py	37	0	8	2	96%	44->47, 54->57
app\crud\users.py	44	0	16	0	100%	
app\database.py	15	4	0	0	73%	17-21
app\main.py	19	2	4	0	91%	34, 38
app\models__init__.py	6	0	0	0	100%	
app\models\book.py	14	0	0	0	100%	
app\models\category.py	9	0	0	0	100%	
app\models\loan.py	13	0	0	0	100%	
app\models\user.py	12	0	0	0	100%	
app\routers__init__.py	0	0	0	0	100%	
app\routers\books.py	30	0	16	0	100%	
app\routers\category.py	12	1	4	0	94%	16
app\routers\loans.py	70	1	36	4	95%	106->109, 128, 142->147, 144->147
app\routers\stats.py	16	0	2	0	100%	
app\utils.py	44	2	12	3	91%	33, 46->48, 53
TOTAL	530	19	138	11	96%	

Coverage HTML written to dir htmlcov

===== 128 passed in 2.27s =====

The screenshot shows the VS Code interface with the Test Explorer open. The left pane displays a list of test cases under the 'ControllerTest' category, including tests for login and register functionality. The right pane shows a detailed view of the 'AuthControllerTest' results, listing individual test methods like 'testLogin_Success()' and 'testRegister_Success()' along with their status (e.g., passed or failed). A sidebar on the right lists all available Java test runner results.

```
%TESTE 5,testLogin_WithEmptyUsername(com.example.Authentication.Auth.AuthControllerTest)
%TESTS 6,testLogin_WithInvalidJson(com.example.Authentication.Auth.AuthControllerTest)

%TESTE 6,testLogin_WithInvalidJson(com.example.Authentication.Auth.AuthControllerTest)

%TESTS 7,testRegister_Success(com.example.Authentication.Auth.AuthControllerTest)

%TESTE 7,testRegister_Success(com.example.Authentication.Auth.AuthControllerTest)

%RUNTIME7479
```

Performance Insights: Stress Testing Results

TEST COVERAGE ANALYSIS BY CRITICAL MODULE

Module	Cov.	Risk	Interpretation
auth_utils.py	95%	Low	Token validation tested
crud/books.py	100%	Low	All CRUD paths covered
crud/loans.py	96%	Low	Loan lifecycle validated
routers/users.py	96%	Low	RBAC well tested
utils.py	91%	Low-Med	Transforms mostly covered
crud/category.py	55%	Med	Basic CRUD only
routers/category.py	94%	Low	Endpoints well tested
routers/loans.py	95%	Low	Endpoints validated

STRESS TESTING RESULTS WITH CONTEXTUAL ANALYSIS

Metric	Value	Interpretation
Total Requests	90	Small concurrent load
Success Rate	57/60 (95%)	Excellent reliability
Failed Auth	3/60 (5%)	Normal variability
Avg Response	4.84ms	Below 50ms target
95th Percentile	8.00ms	Consistent performance
Max Response	52ms	Acceptable worst case
Throughput	22.66 req/s	Supports 100 users

All checks have passed

2 successful checks

CI - E2E (Compose + Behave + JMeter) / e2e (push) Successful in 3m	Details
CI / test-and-build (push) Successful in 1m	Details

APDEX (Application Performance Index)			
Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.667	500 ms	1 sec 500 ms	Total
0.000	500 ms	1 sec 500 ms	GET /users-1
0.000	500 ms	1 sec 500 ms	POST Java /auth/login
0.000	500 ms	1 sec 500 ms	GET /loans
0.000	500 ms	1 sec 500 ms	POST Java /auth/register
0.000	500 ms	1 sec 500 ms	GET /users
0.000	500 ms	1 sec 500 ms	GET /loans-1
1.000	500 ms	1 sec 500 ms	POST /categories
1.000	500 ms	1 sec 500 ms	GET /books
1.000	500 ms	1 sec 500 ms	GET /categories-1
1.000	500 ms	1 sec 500 ms	GET /users-0
1.000	500 ms	1 sec 500 ms	GET /categories-0
1.000	500 ms	1 sec 500 ms	GET /books-0
1.000	500 ms	1 sec 500 ms	GET /books-1
1.000	500 ms	1 sec 500 ms	POST /books
1.000	500 ms	1 sec 500 ms	GET /categories
1.000	500 ms	1 sec 500 ms	GET /books/available
1.000	500 ms	1 sec 500 ms	GET Python /health
1.000	500 ms	1 sec 500 ms	GET /loans-0



Architectural Trade-offs: Retrospective Assessment

Was the dual-service architecture justified for this project?

Realized Advantages

- Security isolation achieved.
- Academic learning objectives met.
- Focused testing simplified.

Realized Challenges

- High operational complexity.
- Data consistency issues.
- Significant development overhead.
- Network latency at scale.

While academically valuable, a monolithic architecture would have been more practical for the project's scale (100 concurrent users).



Conclusion & Future Recommendations

BookWiseUD is functional and robust, but architectural choices highlight key lessons.

1

API Gateway

Implement for simplified frontend and centralized JWT validation.

2

Event-Driven Sync

Use message brokers for user data consistency.

3

Centralized Logging

Essential for production debugging across services.

For similar projects at this scale, a well-designed monolithic backend with DDD alignment is recommended over service separation.



THANK YOU VERY MUCH

