BookWiseUD: A Scalable Library Management System

Presented by:

Wilder Steven Hernandez Manosalva

Jhon Javier Castañeda Alvarado

BookWiseUD

Addressing Library Management Challenges

Manual Systems

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

Lack of Integration

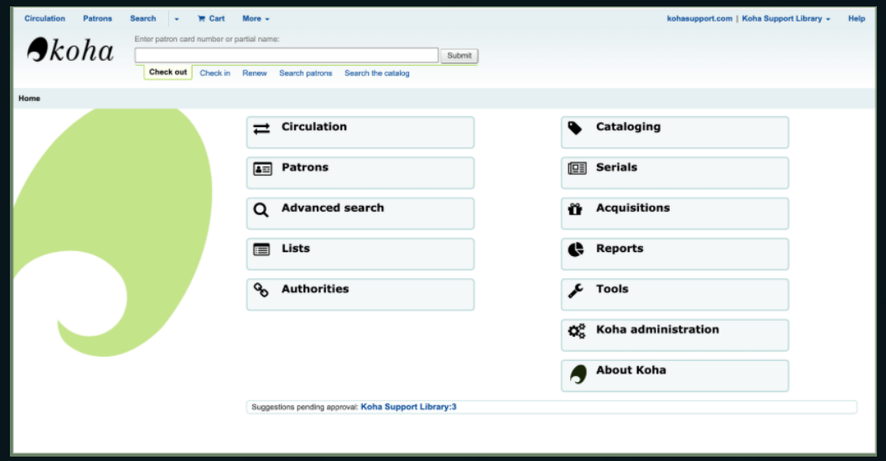
No real-time inventory, automated loans, or secure user authentication.

Poor User Experience

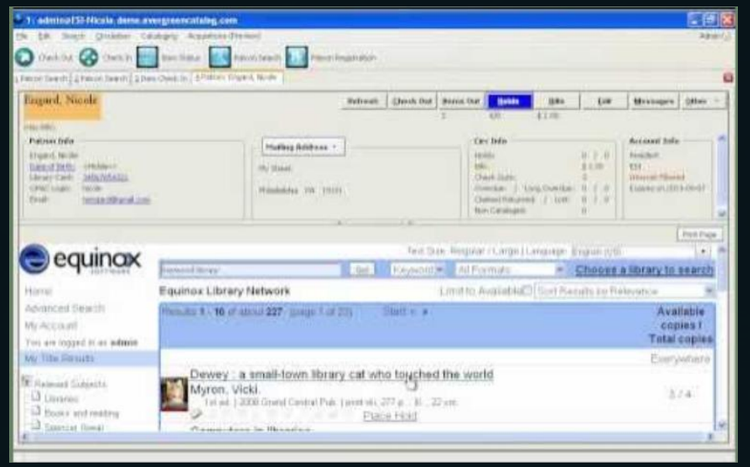
Patrons and administrators face bottlenecks and limited access to information.

Base or references (inspiration)

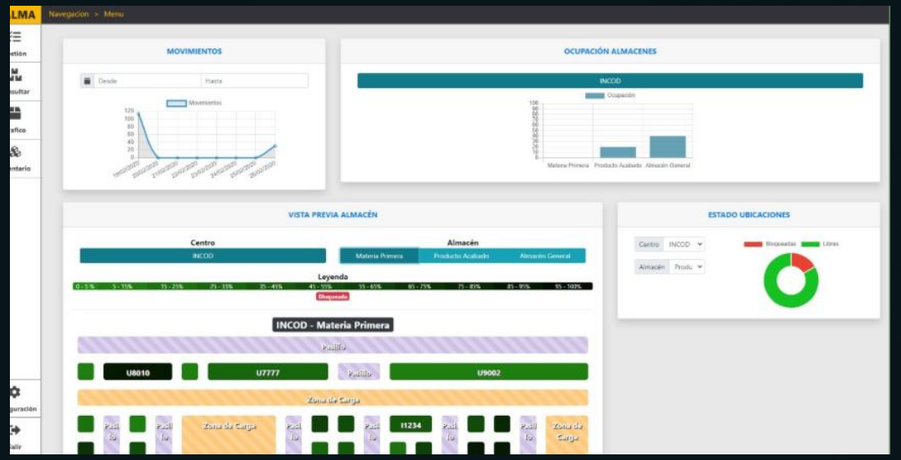
Koha Integrated Library System



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

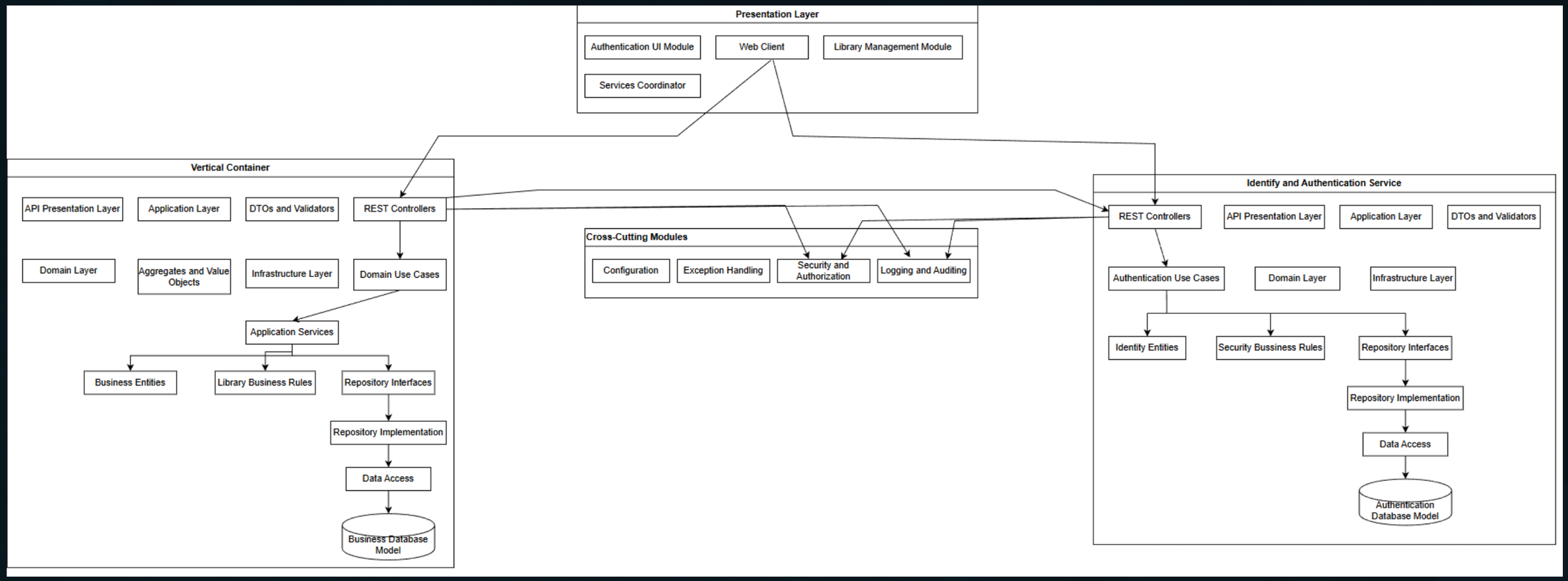
Credential storage separated from domain operations.

Technology Specialization

Spring Security for auth, FastAPI for performance.

Academic Objectives

Demonstrate distributed systems and microservices.



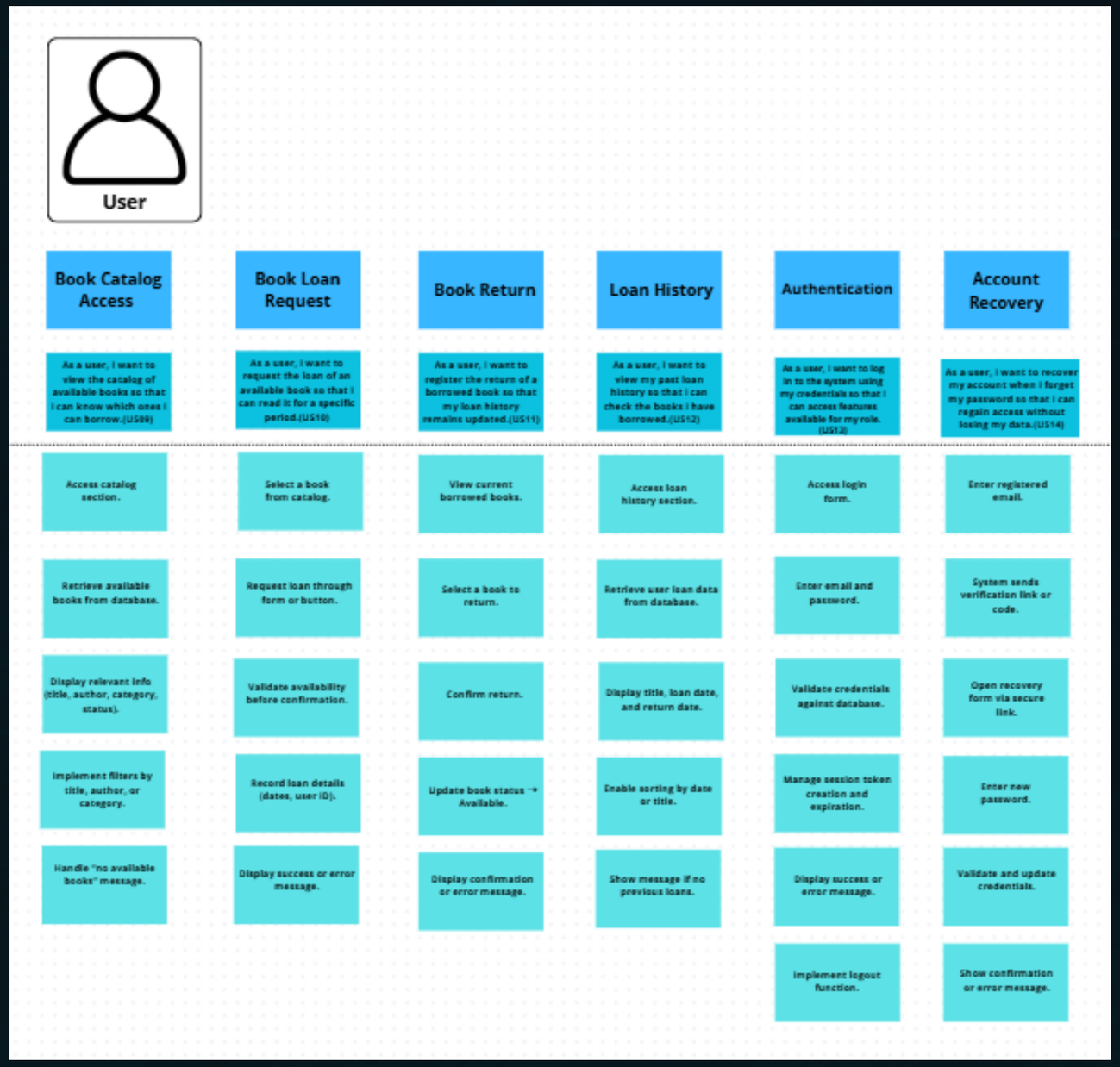
Users Stories

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.

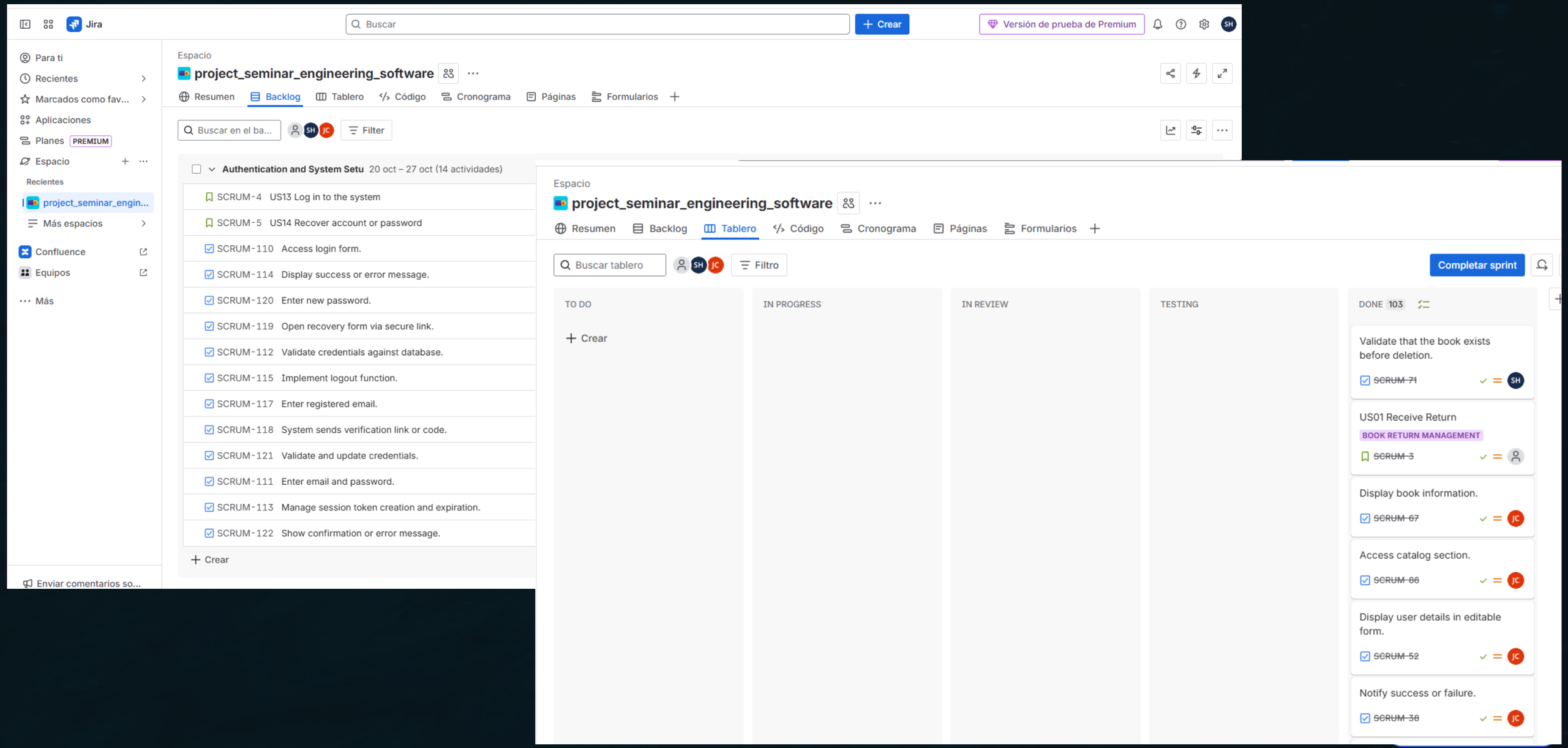
TABLE I
USER STORIES IMPLEMENTATION SUMMARY

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



JIRA


Jira
Buscar
Crear
Versión de prueba de Premium
Para ti
Recientes
Marcados como fav...
Aplicaciones
Planes PREMIUM
Espacio
Recientes
project_seminar_engin...
Más espacios
Confluence
Equipos
Más
Enviar comentarios so...
Espacio
project_seminar_engineering_software
Resumen
Backlog
Tablero
Código
Cronograma
Páginas
Formularios
Buscar en el ba...
Filter
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.
Crear
Espacio
project_seminar_engineering_software
Resumen
Backlog
Tablero
Código
Cronograma
Páginas
Formularios
Buscar tablero
Filtro
Completar sprint
TO DO
Crear
IN PROGRESS
IN REVIEW
TESTING
DONE 103
Validate that the book exists before deletion.
SCRUM-71
US01 Receive Return
BOOK RETURN MANAGEMENT
SCRUM-3
Display book information.
SCRUM-67
Access catalog section.
SCRUM-86
Display user details in editable form.
SCRUM-52
Notify success or failure.
SCRUM-38

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.

```
(venv) PS C:\Users\jjavi\OneDrive\Documentos\Programacion\SpringBoot\Authentication\BookWiseUD\Bc
---------- coverage: platform win32, python 3.12.0-final-0 -----------
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 ==========================
```

```
PROBLEMS 24  OUTPUT  DEBUG CONSOLE  TERMINAL  TEST RESULTS  PORTS
ControllerTest)

%TESTE  5,testLogin_WithEmptyUsername(com.example.Authentication.Auth.Auth
ControllerTest)
%TESTS  6,testLogin_WithInvalidJson(com.example.Authentication.Auth.AuthCo
ntrollerTest)

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

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

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

%RUNTIME7479
```

Test Runner for Java

- AuthControllerTest $(symbol-namespace) com.example.Authenticati...
- testLogin_Success() $(symbol-class) AuthControllerTest ‹ $(symbol-n...
- testLogin_WithEmptyUsername() $(symbol-class) AuthControllerTes...
- testLogin_WithInvalidJson() $(symbol-class) AuthControllerTest ‹ $(s...
- testRegister_ServiceThrowsException() $(symbol-class) AuthContro...
- testRegister_Success() $(symbol-class) AuthControllerTest ‹ $(symbol...

> 8 older results

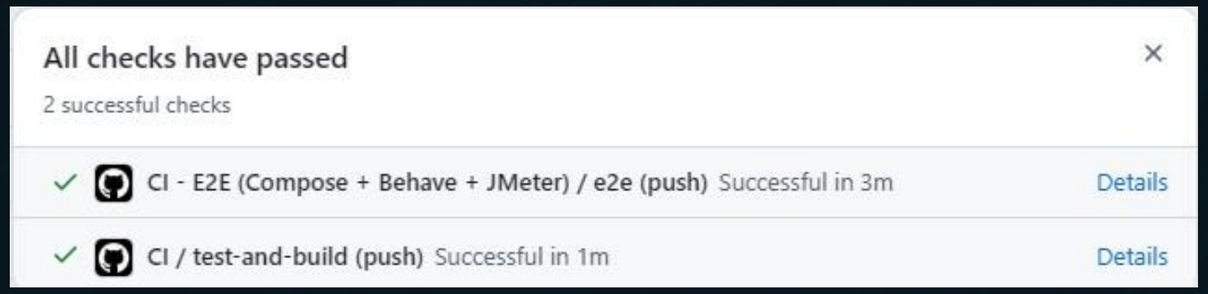
Performance Insights: Stress Testing Results

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

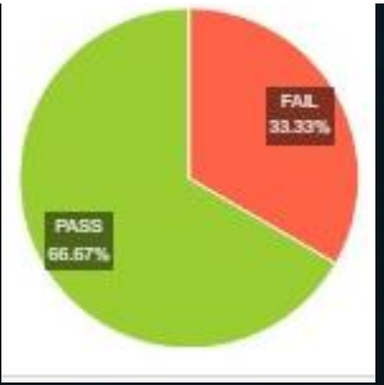
Stress Testing Results with Contextual Analysis

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



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

BookWiseUD