The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 3, June 2025
https://doi.org/10.33022/ijcs.v14i3.4756

A Multi-Tenant Platform for Web-Based Library Applications Using Extreme Programming Methods.

Muhamad Faqih Azhar[1], Sirojul Munir[2], Zaki Imaduddin[3]
muha21092ti@student.nurulfikri.ac.id[1], rojulman@nurulfikri.ac.id[2], zaki@nurulfikri.ac.id[3]
[1,2,3] Department of Informatics Engineering, Nurul Fikri College of Technology, Depok 16451, Indonesia

Article Information

Received : 24 Feb 2025
Revised : 16 May 2025
Accepted : 9 Jun 2025

Keywords

Extreme Programming, Library applications, Multi-tenant.

Abstract

This research aims to develop a web-based platform for book collection lending management with a multi-tenant concept using the Extreme Programming (XP) development method, which includes the frontend and backend parts. This platform is designed to support the needs of library units and book collectors, allowing each entity to operate independently in one system to improve operational efficiency and effectiveness. The Research and Development (R&D) approach used in this research includes the following stages of planning, designing, implementing, and testing within the XP method framework. The technology used to develop this web-based platform includes Golang for the backend and React TypeScript for the frontend, with PostgreSQL as the database. Data was collected through observation and literature studies, while system testing used the Black Box Testing method. This platform provides various main features, such as library or collector registration, book collection management, borrowing and returning processes, and transaction history recording. The study results showed that platform development was carried out in six iterations over three months with a sprint duration of two weeks, resulting in an average work speed per sprint of 13.6 points. The test results showed that all features functioned according to user needs without any functional errors, with a system success rate of 100%. Overall, this platform is stated to be of good quality, ready to operate, and expected to support the management of digital book collections effectively and efficiently.

A. Introduction

Cloud computing-based service models are the solution today, especially with full support for Software as a Service (SaaS) features[1]. Cloud computing provides the flexibility, scalability, and cost efficiency companies need to manage various applications and resources, including multi-tenant applications[2]. Multi-tenant web applications offer features that include authentication, authorization, tenant tracking, and application platform controllers[3]. These systems allow data isolation between tenants through several methods, such as using specialized databases, specialized table schemas, or adding tenant ID columns to shared tables [3].

Public libraries and personal libraries, at the heart of education, are knowledge resources and information centers vital in researching, learning, and developing knowledge of interpretation for individual experiences and communities, socio-cultural contexts, and the formation of disciplines of knowledge[4], [5]. To maximize its role, libraries must adapt to the latest technological developments. One first step is switching from manual to digital systems when managing their collections and services. Manual management is not wrong, but this method is often time-consuming and prone to recording errors, damage, or data loss, which can reduce operational effectiveness and efficiency [6], [7].

Many previous studies have discussed the development of web-based library information systems. However, most of the solutions produced still use a single-tenant concept model, which only allows use by one tenant. [8], [9], [10]. While this model is sound, single-tenant systems are limited in scope and cannot support simultaneous use by multiple libraries or book collectors. Conversely, multi-tenant architecture emerged as an innovative approach to help improve efficiency and scalability. Multitenancy is an architectural style in SaaS that allows various independent users (tenants) to share a single web application instance. This approach provides significant resource efficiency and reduced operational costs by sharing hardware and operating systems, middleware, and application components[11].

Khairul Fahmi (2016) conducted previous research that developed a multi-tenant-based library information system to build a multi-tenant-based library application for schools that do not have sufficient finances[12]. This system allows several libraries to share one platform with separate access for each tenant, thereby increasing the efficiency of digital library management. However, the technology used in the study still uses old technology and is not entirely optimal regarding flexibility, scalability, and ease of integration with other systems, and its appearance is not fresh.

In developing a web-based multi-tenant platform, two main components, the front and backend, have complementary roles. The frontend is the system part that directly interacts with users, including the visual interface and elements that users can access directly[13]. Its function is to present data in a way that is easy to understand and allows users to interact with the system. On the other hand, the backend is the part that works behind the scenes to handle business logic, process requests, and manage and store data [13]. The backend ensures the data received from the frontend is processed correctly and returned according to user needs.

This research aims to design and develop a web-based library management system with a multi-tenant concept. Not only libraries this platform also allows book collectors to use it independently. Developing this multi-tenant platform can increase operational efficiency and improve user service quality. The goal of this research is to boost the efficiency of tenant management by putting in place elements that facilitate multi-tenant business processes and allow each tenant to function independently without affecting the other tenants.

In this research, React TypeScript is used to build the frontend. React supports a component-based architecture approach that breaks down interfaces into small components that are independent, modular, and reusable [13]. This approach simplifies system development, testing, and maintenance. Additionally, React makes it possible to construct single-page applications (SPAs), which offer a quicker user experience because only a section of the page needs to be updated. Meanwhile, the backend side will be handled using Golang. Golang is used to build the backend because it is superior in handling parallel processes and large-scale requests[14]. The backend is designed with a RESTful API to ensure smooth frontend integration.

B. Method

This study was carried out in phases, as Figure 1 illustrates. The first stage begins with observation and literature review with output in the form of related research information and understanding of business processes through literature review. The second phase involves doing a business process analysis to identify the systems that need to be put into place, followed by the design and implementation stage using the extreme programming (XP) method, followed by an evaluation and the formulation of conclusions.

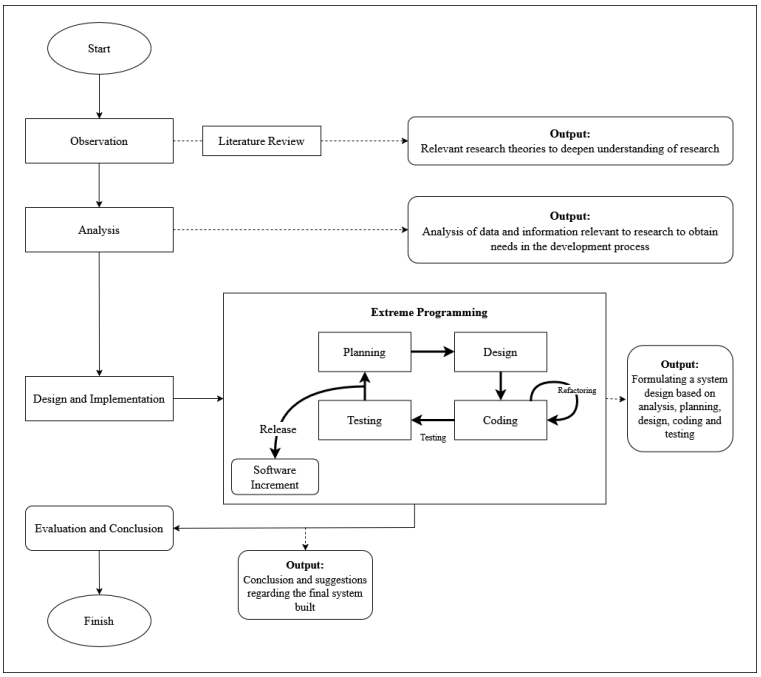

Figure 1. The Flow of Research Stages

XP is a software development method that emphasizes rapid iteration, close team collaboration, and the ability to adapt to changing needs. This method emphasizes communication, feedback, simplicity, boldness, and respect. With a short and iterative development cycle, XP makes it easier to manage complex projects, ensuring that each stage produces concrete output that can be evaluated from the start [15]. The process begins with planning to understand the relevant theory, followed by system design and mockups as needed. Furthermore, implementation is carried out through backend and frontend coding, followed by testing to ensure that the features function optimally. The final stage concludes the entire process, which makes the platform ready for use. The following are the details of each stage:

1. Observation
   This stage involves gathering information and literature to understand system requirements, formulate objectives, select technology, and identify potential constraints during system development.
2. Analysis
   The results of the observations that have been carried out are then analyzed to obtain an outline of the problem and prepare materials for design.
3. Design and Implementation
   This stage uses the XP method to create fast and flexible software development. This methodology includes several stages, namely planning, design, coding, and Testing[15], [16]. The following explains each stage:
   a. Planning
      During this stage, researchers conduct a needs analysis based on data gathered in prior stages. This analysis helps researchers understand the system's workflow, define the necessary features, and provide a clear overview of the expected outcomes.
   b. Design
      Researchers create a system model in the design phase based on the needs analysis results. A database model is also developed to illustrate the relationships between data within the system. Researchers utilize the Unified Modeling Language (UML) for system modeling, including diagrams such as Use Case Diagrams, Activity Diagrams, and Deployment Diagrams. For database modeling, an Entity-Relationship Diagram (ERD) is used[16].
   c. Coding
      The coding stage involves implementing the system model designed in the previous phase. The design is translated into program code using the Golang programming language for backend development and React TypeScript for the frontend. The database implementation uses PostgreSQL to manage data storage and retrieval effectively.
   d. Testing
      This stage involves testing the system to ensure its features and functionalities meet user requirements. User feedback is incorporated into the testing process. The Black Box testing method focuses on system inputs and outputs without delving into the system's internal workings.
4. Evaluation and Conclusion

This stage is the end of the development process, which aims to evaluate the success of the system that has been built. Conclusions are made based on the results of system testing. If the system successfully meets the needs and objectives formulated in the initial stage, then the development is declared successful. However, if there are deficiencies, recommendations will be recorded for further development or future improvements. Thus, this stage assesses the project's success and becomes the basis for future iterations or system improvements.

C. Result and Discussion

This stage is divided into two main phases: analysis, design, implementation, and testing. The following are the details of each stage.

1. Analysis and Design

The first stage's result is the system analysis and design process. This stage explains in detail the system analysis process and design steps that are the primary basis for developing a multi-tenant-based book lending platform. The explanation focuses on identifying system requirements, modeling workflows, and technical design to ensure the application meets research objectives and user needs. The results of this analysis will first be mapped into a user requirement table.

Table 1. User Requirements

Features	Super Admin	*Owner*	*User*
Access Login and Account Registration	✓	✓	✓
Managing Profile Data	✓	✓	✓
Creating a New Tenant			✓
Viewing Collection List and Details	✓	✓	✓
Making Transactions			✓
Viewing Transaction History		✓	✓
Managing Collection/Tenant Data		✓	
Managing Genre Data	✓		
Managing Category Data	✓		
Deleting Collections	✓		
Deleting Users	✓		
Deleting Tenant	✓		

In Table 1, User Requirements, it can be seen that each role in the platform has different access rights to system features. Super Admin has full access to manage all aspects of the system, including upgrading the user role to Super Admin, managing collection data, genres, and categories, and deleting collections, users, and libraries. The Owner/Admin (Manager) has more limited access rights than the Super Admin but includes managerial features such as tenant profile management, collection management, and access to view collection lists and details. Managers can also make transactions and view transaction history. Then, Users (Visitors) have limited access to essential activities such as account

registration, viewing collections, and creating and monitoring transactions. All roles, including Super Admin and Owner/Admin, were initially Users who could then be upgraded to higher roles. The results of these user requirements were then made into a user story and system development plan by creating development iterations using the XP method. This iteration was carried out 6 times over 3 months. Details of the iteration implementation can be seen in Table 2 below.

Table 2. User Stories and XP Sprint/Iterations

Sprint	Task	Role	User Story	Story Point
1	Auth	All Role	I want to register a new account and log in	5
	Register new Tenant	User/ Visitors	I want to register new tenants and manage my collections.	5
	Manage Collection	Owner/ Manager	I want to manage the collections I have (CRUD)	5
2	Manage Collections	Super Admin	I want to manage a collection of all tenants (see details and delete if they don't match the criteria)	5
	Collection Transactions	User/ Visitors	I want to see the collections and collection details from various tenants.	3
		Owner/ Manager	I want to borrow and return a collection from a tenant.	5
3	Transaction History	User/ Visitors	I want to see the data and details of the transaction history that I have made.	5
		Owner/ Manager	I want to see data and details of transaction history made at this tenant.	5
	Manage Transactions	Owner/ Manager	I want to monitor data on all outstanding loans	5
4	Manage Transactions	Owner/ Manager	I want to add new loan data	8
			I want to mark a loan as completed and the collection returned.	5
5	Manage Users	Super Admin	I want to manage user data registered on the platform (view details and delete if invalid and upgrade roles if necessary)	5
	Manage Genres		I want to manage the collection of genre data (CRUD)	8
6	Manage Categories	Super Admin	I want to manage collection category data (CRUD)	8
	Manage Tenants		I want to manage tenant data (view details and delete them if necessary)	5

The next stage of the results is to analyze the platform needs using use case diagrams. Use case diagrams to describe the interaction between actors and the system being developed to model its main functionality or scenarios.[15]. For the multi-tenant book lending platform's use case diagram, the interaction occurs between the actor and the system. The actors involved include Users, Owner, and Super Admin. Each actor must first log in to gain access according to their role. Users can view and borrow collections and view transaction history. The owner is responsible for tenant operations and can access collections, transactions, and

tenant profiles. Super Admin has full access rights, including managing the system, upgrading user roles, and managing categories, genres, and tenant data. The detailed diagram can be seen in Figure 2 below.

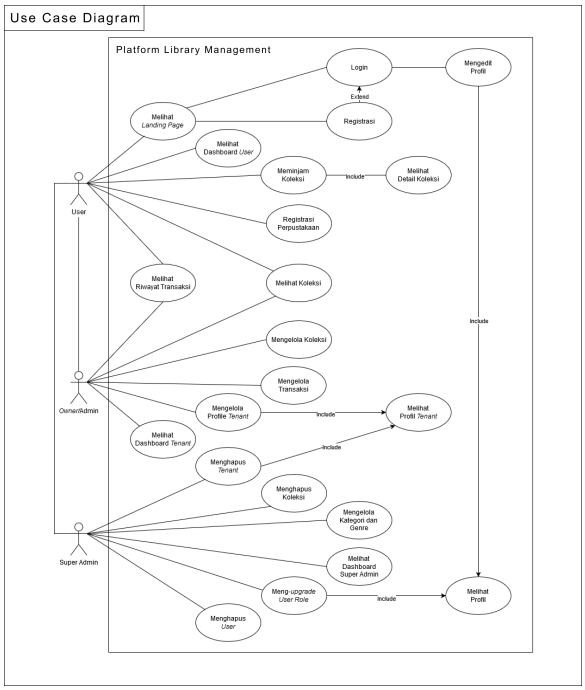

Figure 2. Use Case Diagram

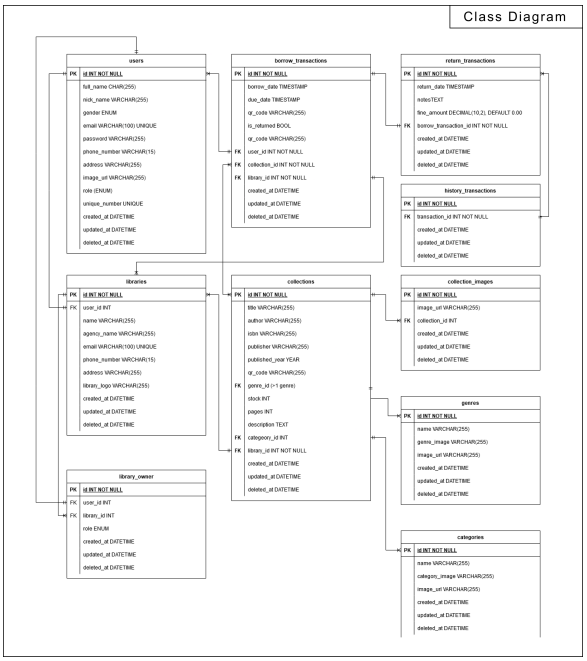

Figure 3. Database Schema

The results of the next stage enter the design stage. The database design is carried out at this stage as an Entity Relationship Diagram (ERD), also known as a class diagram. A class diagram is a diagram that displays the structure of a system by describing classes, attributes, methods, and relationships between classes[15].

Figure 3 above depicts the database schema of the multi-tenant book lending platform. The image also shows several main entities in the system: Users, Libraries, Genres, Collection_Images, Collections, Library_Owner, Categories, Borrow_Transactions, Return_Transactions, and History_Transactions. Each entity has several attributes appropriate for storing related data.

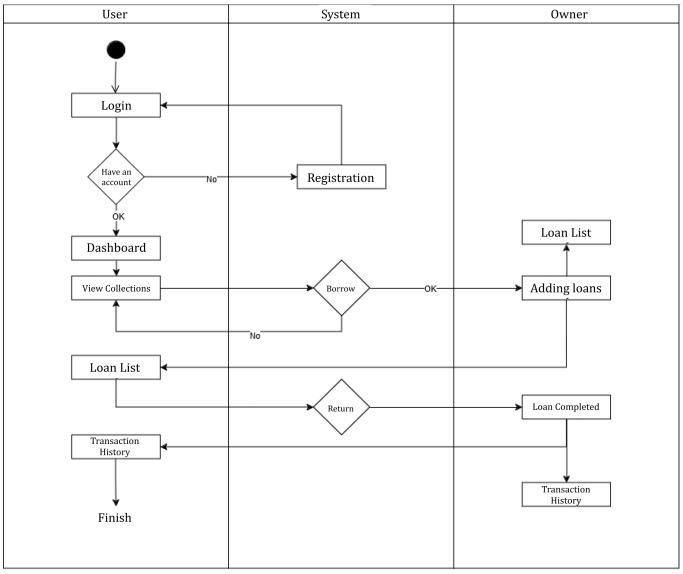

Figure 4. Collection Borrowing Activity Diagram

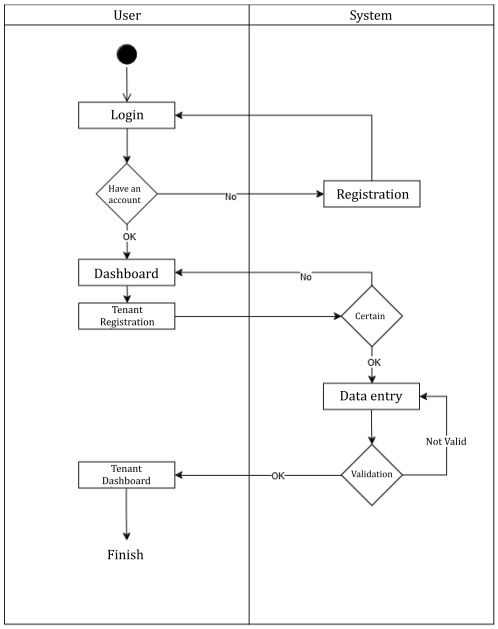

Figure 5. Activity Diagram New Tenant Registration

The following results are mapped into an activity diagram. This activity diagram displays the system's process flow or activities[15]. Activity diagrams are often used to visualize workflows, business logic, and decision-making conditions[17]. Figures 4 and 5 above show activity diagrams of the central business processes on this platform, namely the collection borrowing activity diagram and tenant creation.

2. Implementation and Testing

The system implementation stage includes the development of a multi-tenant book lending platform based on the analysis and design stages carried out previously. This section will display the results of the platform interface implementation for its two main features, namely the collection lending feature, as shown in Figure 6, and the collection management feature, as shown in Figure 7.

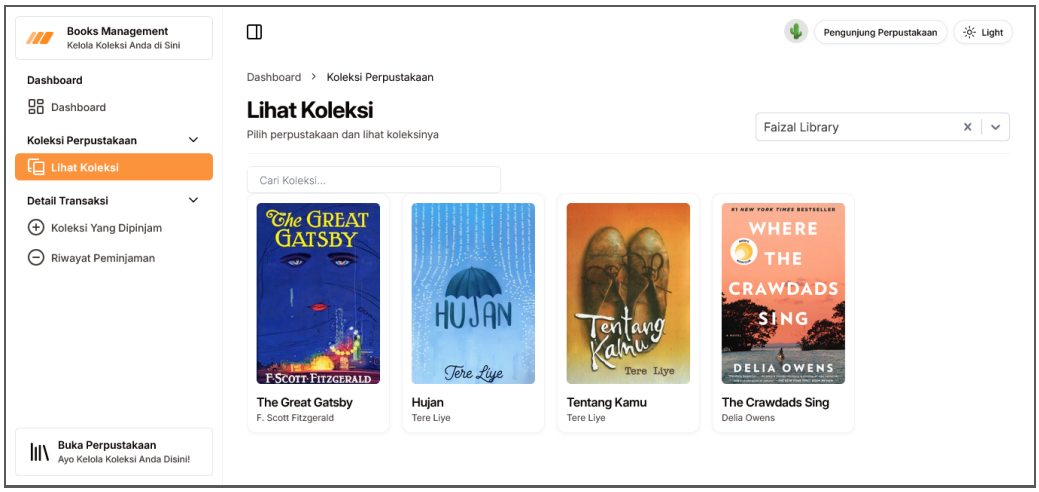

Figure 6 Lending Features Dashboard

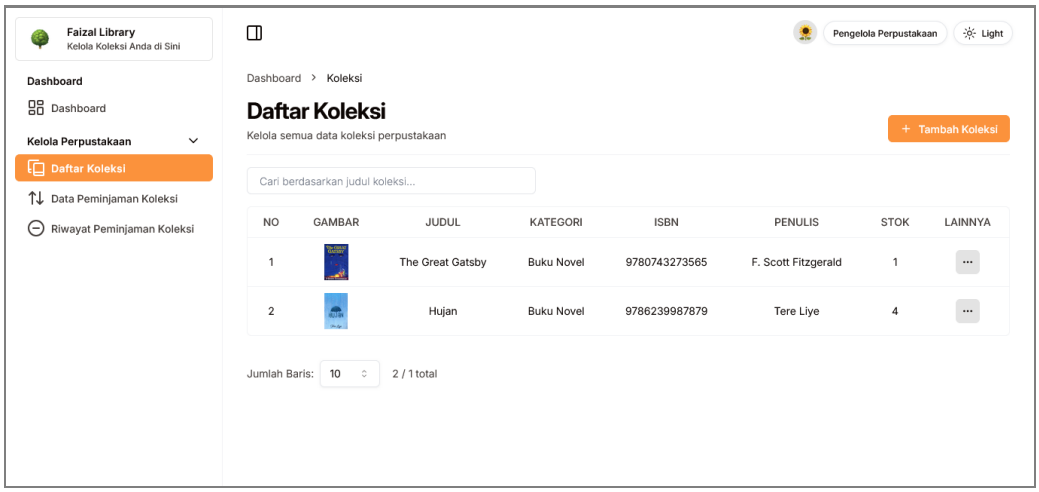

Figure 7 Manage Collections Dashboard

The black box testing method was used for this study. Black box testing is a software testing method focusing on external functions without considering its internal structure[18]. This approach aims to identify various types of errors, such as features that do not work or are missing, problems with the structure and database, data access errors, and issues related to the interface, user experience, performance, initialization, and system termination[19]. All Multi-Tenant Book Lending Platform features have been tested based on previously designed scenarios. All significant features were successfully run without any failures, resulting in a success rate of 100%. Table 3 below maps the results of testing on several major features on the platform.

Figure 8. Black Box Testing Results On Key Features

Task	Scenario	Action	Expected Result	Result
Auth	New users can register and login	Fill in all registration and login data completely and validly	The user successfully registered and logged in	Valid
Register new Tenant	Users can register new tenants	Fill in all tenant registration data completely and validly	The user successfully registered a new tenant	Valid
Manage Collection	Owner/Manager can manage collection data	Fill in all data completely and validly	The system successfully processed the management of collection data (CRUD)	Valid
	Super Admin can manage all tenant collections	View or delete collections from each tenant	The system successfully displays and deletes collection details from a selected tenant	Valid
Collection Transactions	Users/Visitors can view the collection and its details.	View the collection from the selected tenant and see the details of the collection	The system displays collection data along with details from the selected tenant	Valid
	Owner/Manager can manage borrowing and returning collection data	Carrying out borrowing and returning collections	The system successfully carries out the process of borrowing and returning collections.	Valid
Transaction History	Users/Visitors can view personal transaction history data	View the transaction history page	The system displays user transaction history data.	Valid
	Owner/Manager can view historical data of all transactions at the tenant	View the transaction history page	The system displays tenant transaction history data.	Valid
Manage Transactions	Owner/Manager can manage transaction data	View, edit, and delete transactions as needed	The system successfully displays details, edits and deletes ongoing transaction data	Valid
Manage Users	Super Admin can manage all users	View or delete users according to the selected data	The system successfully displays and deletes the selected user details.	Valid
Manage Genres and Categories	Super Admin can manage genre and category data	Fill in all data completely and validly	The system successfully processed the management of genre and category data (CRUD)	Valid
Manage Tenants	Super Admin can manage all tenants	View or delete a tenant	The system successfully displays	Valid

	details and deletes the selected tenant.

D. Conclusion

This research successfully designed and developed a web-based library management system with a multi-tenant concept. This system allows each tenant to manage collections, loans, and transactions independently. The development was carried out using the Extreme Programming method, which consisted of six sprints over three months, including needs analysis and backend and frontend implementation. The speed of work points in sprints one to six were 15 points, 13 points, 15 points, 13 points, 13 points, and 13 points. If the average value is drawn, each sprint's work speed is 13.6 points.

The leading technologies used in the development include Golang for the backend, React for the frontend, and PostgreSQL for the database. The resulting platform effectively supports multi-tenant business processes with features tailored to each user: User, Owner/Admin (manager), and Super Admin. This platform allows each tenant to manage its operations independently, including book collection management, loan transactions, and loan history. Meanwhile, the Super Admin plays a role in managing global data such as categories, genres, and tenant management so that the system can run in a structured manner. Based on testing using the Black Box Testing method, all system features function well without failure, with a success rate of 100%. This shows that the platform that has been developed has met the designed specifications and can support multi-tenant operations effectively and optimally.

E. Acknowledgment

I want to express my gratitude to all parties who have provided support for this research. My special thanks go to our colleagues at Sekolah Tinggi Teknologi Terpadu Nurul Fikri for their invaluable input and encouragement.

F. References

[1] A. Abdi and S. R. M. Zeebaree, “Embracing Distributed Systems for Efficient Cloud Resource Management: A Review of Techniques and Methodologies,” *Indones. J. Comput. Sci.*, vol. 13, no. 2, pp. 1912–1933, 2024.

[2] P. Czarnecka, “The Multi-Tenant Cloud Computing Architecture Allows the Service Consumers to Share the Computing,” *Tennessee Res. Int. Soc. Sci.*, vol. 2, no. March, pp. 1–24, 2020.

[3] S. K. Pippal, S. Kumar, and R. Rani, “Optimizing multi-tenant database architecture for efficient software as a service delivery,” *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 22, no. 5, pp. 1128–1137, 2024.

[4] J. Tarango, J. D. Machin-Mastromatteo, and J. Cortés-Vera, “Chapter sixteen - Information users as active prosumers: perspectives from social marketing and sociocultural value for academic libraries’ benchmarking processes,” D. Baker, L. Ellis, C. Williams, and C. B. T.-B. L. Wragg Information and Education Services, Eds., Chandos Publishing, 2023, pp. 265–279.

[5] A. Hedley and B. Magazines, “Papers of the Bibliographical Society of Canada

60 (2023),” vol. 60, pp. 2021–2024, 2023.
[6] M. Lamada, F. Fathahillah, A. Zakilah Ifani, and F. Wahyu, “Analisis Kualitas dan Pengembangan Sistem Informasi Perpustakaan berbasis web menggunakan Teknologi Barcode,” *J. Embed. Syst. Secur. Intell. Syst.*, vol. 3, no. 1, p. 15, 2022.
[7] V. Anjelia, A. Rahman, and D. Destiarini, “Rancang Bangun Sistem Informasi Perpustakaan Berbasis Web Pada Pada Mts Al – Husna Depok,” *Intech*, vol. 4, no. 1, pp. 7–12, 2023.
[8] S. J. Ayano, S. A. Ojomu, G. Fisher, and E. O. Onoyom-, “Developing a Library Management System Utilizing Python,” vol. 7, no. 6, pp. 318–322, 2023.
[9] A. Mulyana, D. Rohpandi, E. B. Sambani, M. Fahmi, N. Sudarsono, and M. Hamdani, “Web-Based Library Information System Design,” *Informatics Manag. Eng. Inf. Syst. J.*, vol. 1, no. 1, pp. 77–87, 2023.
[10] W. N. Fadhilah and M. Maryam, “Development of Library Information System Web-based of SMA Negeri 1 Mojolaban Sukoharjo,” *Emit. J. Tek. Elektro*, vol. 21, no. 2, pp. 78–86, 2021.
[11] I. Kumara, J. Han, A. Colman, W. J. van den Heuvel, D. A. Tamburri, and M. Kapuruge, “SDSN@RT: A middleware environment for single-instance multitenant cloud applications,” *Softw. - Pract. Exp.*, vol. 49, no. 5, pp. 813–839, 2020.
[12] K. Fahmi, A. T. Haryono, I. F. Astuti, and D. Cahyadi, “Perancangan Dan Implementasi Aplikasi Perpustakaan Berbasis Multitenant,” *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 11, no. 1, p. 1, 2016.
[13] L. Kurapati, “Different Types of Architectures in Frontend Design and Development,” *Int. J. Multidiscip. Res.*, vol. 6, no. 5, pp. 1–5, 2024.
[14] R. M. Isnaeni, N. I. Utama, and S. Suakanto, “Backend Development of a Microservice-Based Website Application for Public Issue Reporting: Case Studyn in People Representative Council,” *J. La Multiapp*, vol. 5, no. 2, pp. 63–77, 2024.
[15] S. Munir, I. Haromain, R. Wahyudi, M. Asqia, and R. Raafi’udin, “Wikuliner - Regional Culinary Recommendation System Based on the Web Using Extreme Programming Method,” *Proc. - 3rd Int. Conf. Informatics, Multimedia, Cyber, Inf. Syst. ICIMCIS 2021*, pp. 102–107, 2021.
[16] H. A. A. Misna Asqia, “Designing a Presence Information System for Student Mentoring Activities Using the Laravel Framework at STT NF,” *Indones. J. Comput. Sci.*, vol. 13, no. 5, pp. 284–301, 2024.
[17] E. Elis and A. Voutama, “Pemanfaatan Uml (Unified Modeling Language) Dalam Perencanaan Sistem Penyewaan Baju Adat Berbasis Website,” *Informatika*, vol. 14, no. 2, p. 26, 2022.
[18] N. Nasrul, H. Saptono, E. Wibowo, and A. Amalia, “Rancang Bangun Sistem Informasi Manajemen Aset Berbasis Web untuk Menghitung Penyusutan Fiskal,” *J. Inform. Terpadu*, vol. 10, no. 1, pp. 66–72, 2024.
[19] E. Novalia and A. Voutama, “Black Box Testing dengan Teknik Equivalence Partitions Pada Aplikasi Android M-Magazine Mading Sekolah,” *Syntax J. Inform.*, vol. 11, no. 01, pp. 23–35, 2022.