APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 5 No. 2 (2024) 1160-1179
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Bookpanda and Grabbook Management System

Nur Aina Syazwina Zainudin[1], Mohd Zanes Sahid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: zanes@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2024.05.02.061

Article Info

Received: 14 July 2024
Accepted: 30 October 2024
Available online: 15 December 2024

Keywords

Box -Based Borrowing, Library Management System, Prototyping, Web-Based System

Abstract

The BookPanda and GrabBook Management System is a web-based platform designed to address school library management challenges. It resolves issues such as borrowing management, data mismanagement, and duplication, boosting administrative productivity. The system eases the administrator's burden with library prefects assisting in borrowing operations, while students can review fines and borrowing records. The technology combines AI, NLP, and machine learning to develop an intelligent chatbot for enhanced user engagement. A recommendation system utilizing the Cosine Similarity algorithm is integrated via Flask, offering customized book suggestions. The prototyping methodology ensured a user-centric design, resulting in a practical and high-quality web-based system developed using Visual Studio Code and a MySQL database. Accessible from library helpdesk computers, it significantly benefits administrators, the library, and students, particularly during peak hours, enhancing the user experience. This system aims to automate tasks, save time, and improve library management efficiency, focusing on scalability and enhanced user services.

1. Introduction

In today's digital age, libraries must adapt to information technology's rapid expansion and universal growth. Hence, to optimize retrieving and accessing information, librarians must establish conducive environments by implementing effective management techniques [1]. Sk Kamunting is a primary school located at Kamunting, Perak. The school library was built later in October 1988. The current borrowing system used is *Automasi Pusat Sumber Interaktif* (APSI). Teacher Aishah, the librarian at SK Kamunting, introduced BookPanda and GrabBook to boost students' reading interests. This innovation has been implemented in some schools in Malaysia and approved by YB Ir Fadhil Nuruddin. BookPanda and GrabBook is a new packaging type of book borrowing based on box. There are two options for this box green for GrabBook and pink for BookPanda. Many books will be inserted into these boxes for students to borrow, and students usually place them in the classroom for their classmates to read. Despite the different colours, both boxes share identical functionalities, catering to the same target audience and offering the same borrowing and usage process.

However, this latest innovation still uses manual methods, which are logbooks and paper, to record and manage borrowing operations. This manual approach has caused various issues, including challenges in managing and sorting the borrowed items, separate systems for book and box borrowing, data mismanagement and duplication. These issues have significantly reduced the efficiency of the library management system, making it time-consuming and tedious. Hence, to overcome this problem, a web-based system called BookPanda and GrabBook Management System will be developed to increase the efficiency of the SK Kamunting library

management system. The objectives of this study are to analyze and design BookPanda and GrabBook Management system using an object-oriented approach, develop a web-based platform for BookPanda and GrabBook Management System and evaluate the functionality of the system using user acceptance testing.

The BookPanda and GrabBook Management System at SK Kamunting Library aims to integrate computerised and manual systems into a unified web-based platform. This integration will significantly enhance the library's efficiency and productivity. Accessible to administrator, students and library prefects, the system will alleviate administrative burdens by granting library prefects access to manage borrowing boxes and book modules. This system consists of fifteen modules, which are login, registration, manage user, manage book, manage box, manage book borrow, manage box borrow, manage fine, manage review, reporting, check fine, check borrowing, chatbot, manage wishlist and manage quote which offers comprehensive functionality while refine library operations.

The article is divided into five main sections. The first section delves into the project's contextual background. The second part focuses on reviewing related literature and existing systems. The third section outlines the methodology, encompassing system analysis and design. The fourth section is result and discussion. The last section is the summary of the proposed system.

2. Literature review

This section will discuss the related concept of BookPanda and GrabBook Management System and comparison with similar system.

2.1 Computerised Information System

A computerised information system refers to a social system that contains various human activities involving the utilisation of computer technology [2]. It is achieved through data collection, storage, processing, and dissemination. Computerised information systems can be classified into various categories, such as desktop-based, mobile-based and web-based. A web-based application can be defined as an information system accessible through a web browser via an internet connection, such as the Internet or an intranet [3]. The BookPanda and GrabBook management system utilize a web-based computerised information system, emphasizing variety and connectivity to enhance user interaction.

2.2 Web-Based System

Web is usually connected to the internet and runs on web browser software such as Firefox, Google Chrome or Microsoft Edge. [4] Web-based apps are extensively utilised by diverse entities, such as individuals, groups, organisations, and governments, to optimise information dissemination and expedite corporate operations [5]. The client-server architecture is a conceptual framework used in distributed systems to illustrate how data and processing activities are spread among different components. The proposed system will be implemented in a web-based system using programming language such as JavaScript, HTML, PHP, and CSS with MYSQL as the database.

HTML, known as Hypertext Markup Language, is used as web browsers' primary markup language to display content [6]. Hypertext Markup Language (HTML) is the computer format used to create websites. A straightforward system that incorporates stylistic elements into web publications, such as fonts, colours, and spacing, is known as Cascading Style Sheets (CSS) [7]. PHP, one of the most widely used programming languages, is frequently employed by developers to construct web-based client-server applications.

JavaScript is widely used as the predominant scripting language for client-side web development. JavaScript is renowned primarily because of its extensive usage and accessibility [8]. MySQL is a popular client/server relational database management system that originated in Scandinavia and use SQL [9].

2.3 Library Management System

The Library Management System is a project designed to create an automated and computerised solution for library operations. Its purpose is to streamline and enhance the management and monitoring of daily library chores, ensuring efficiency and effectiveness [10]. Hence, the library management system will facilitate the task of librarians. The proposed system falls under the category of a primary library management system

2.4 Study of Existing System

An analysis was conducted to compare three existing library management systems, namely S-LIB, EZYPSS, and PREPUS, in order to identify their strengths and weaknesses in relation to the current system. Table 1 presents a comparison between the current system and the proposed system.

Table 1 *Comparison with Existing System*

Features/System	S-LIB	EZYPSS	PREPUS	BookPanda and GrabBook Management System
Log In	Yes (Administrator)	Yes (Administrator)	Yes (Administrator, library prefect)	Yes (Administrator, library prefect, student)
Registration	Yes (Administrator)	Yes (Administrator)	Yes (Administrator)	Yes (Administrator, library prefect. Only administrator can register new administrator)
Manage User	No	No	Yes (Administrator can edit and delete)	Yes (Administrator can edit and view)
Manage Book	Yes (Administrator can create, read, edit and delete)	Yes (Administrator can create, read, edit and delete)	Yes (Administrator can create, read, edit and delete)	Yes (Administrator can create, read, edit and delete)
Manage Box	No	No	No	Yes (Administrator can create, read, edit and delete)
System User	Administrator	Administrator	Administrator and library prefect	Administrator, student and library prefect
Check Borrowing	No	No	No	Yes (Student, library prefect)
ChatBot	No	No	No	Yes (Student, library prefect)
Manage Box Borrow	No	No	No	Yes (Scan/ insert details. Available for administrator and library prefect)
Manage Book Borrow	Yes (Scan/ insert details. Available for administrator and library prefect)	Yes (Scan/ insert details. Available for administrator and library prefect	Yes (Scan/ insert details. Available for administrator and library prefect	Yes (Scan/ insert details. Available for administrator and library prefect)

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

Table 1: *(cont)*

Manage Review	No	No	No	Yes (Administrator, student, library prefect)
Manage Report	Yes (Report generated based on administrator chosen category)	Yes (Report generated based on administrator chosen category)	Yes (Report generated based on administrator chosen category)	Yes (Report generated based on administrator chosen category)
Manage Fine	Yes (Administrator can see fine record and accept payment)	Yes (Administrator can see fine record and accept payment)	Yes (Administrator can see fine record and accept payment)	Yes (Administrator can see fine record, issue fine and accept payment)
Manage Wishlist	No	No	No	Yes (Student, library prefect)
Manage Quote	No	No	No	Yes (Student, library prefect, administrator)
Check Fine	No	No	No	Yes (Student, library prefect)

Upon examination, all systems share five common modules which are login, registration, manage book, manage book borrow, manage fine, and manage report. However, the proposed system features two notable distinctions. In the proposed system, the login module is accessible for students, library prefects, and administrators, while the registration module is available for both library prefects and administrators. A standout feature of this proposed system is its box-based book borrowing system, enabling students to borrow books in bulk. This innovative feature allows students to borrow multiple books simultaneously under one issuance record. There is no system available for this system since it is a new innovation implemented to increase reading interest among students. Additionally, the existing systems lack modules such as managing box borrows, checking fines, checking borrows, chatbots, manage quote, manage wishlist and managing reviews, thus highlighting the uniqueness of the proposed system.

3. Methodology

The Prototype methodology is adopted for the proposed system, expediting the creation of an improved version presented to users for evaluation and feedback [11]. The chosen prototype is system prototyping. This approach actively engages users in development, allowing ongoing modifications to enhance the system's quality. Chosen for its effectiveness in addressing user-expressed system needs, stakeholders supported its use to expand the system with more users and modules.

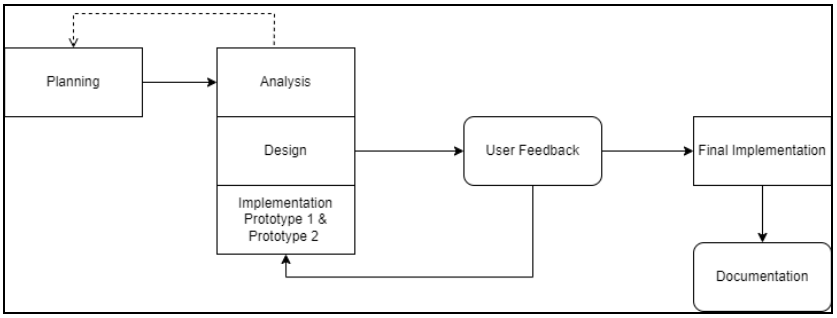

Fig.1 *System Prototyping Model [12]*

The prototype model comprises six discrete stages, namely planning, analysis, design, implementation prototype, final implementation system, and testing. Table 2 exhibits the association between each phase of the project development process and the relevant tasks and deliverables that must be completed.

Table 2 *System Development Workflow*

Phase	Task	Output
Planning	• Decide and identify the proposed title. • Identify the issue faced by the library school. • Understand the currently existing system workflow. • Understand stakeholder requirement. • Determine the system objectives, goal and scope	• Project Proposal • Develop Gantt Chart • List of the objectives, problem statement and scope
Analysis	• Analysis the existing system workflow • Create an AS-IS model for the existing system workflow. • Identify, compare and analyse differences between the proposed system and the existing system. • Determine the system users and functions. • Create use case specification. • Draw the use case, activity diagram, class diagram and sequence diagram. • Identify the system hardware requirements	• AS-IS model diagram for the existing system • Interview Questions • Comparison table with the related existing systems • Use Case (To-Be) • Activity diagram • Class diagram
Design	• Create an initial prototype for the proposed system • Design system interface and user interface based on user requirements • Review Prototype one and made a design based on user requirements.	• User Interface Design • System Interface
Implementation Prototype	• Develop prototype one for iteration one and prototype two for iteration two. • Test the prototype one and two. • Present the prototype to the stakeholder. • Made a modification to the prototype based on user requirements and feedback. • Coding the program using the JavaScript, HTML, PHP and CSS.	• Prototype One • Prototype Two
Implementation System	• Develop system based on the accepted final prototype. • Ensures that all of the system's features and functionalities are implemented. • Review the system with stakeholder.	• System Prototype
Testing	• Create test cases • Test the system to detect the defects and bugs of the system. • Conduct a user acceptance testing. • Conduct functional and non-functional testing. • Require stakeholder to test the system to ensure the system fulfil the requirements	• Test Report • System

3.1 Requirement

A requirement, is a concise declaration of the essential functionalities or attributes a system must possess [13]. During the analysis phase, the requirements are derived from a business perspective, focusing on the essential

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

functionalities needed to fulfil user expectations. A functional requirement defines the fundamental characteristics a system should possess. Table 3 display the functional requirements for the systems.

Table 3 *Functional Requirements*

Module	Description
Login	• The system shall only allow username and password that had been verified by the system to accessed the system. • The system shall allow respective user to only access their own interface. • The system shall display error message if user fail to insert correct username or password.
Register	• The system should allow administrator to register new student, library prefect and administrator into the system. • The system should allow library prefects to register only new students and library prefects into the system. • The system should ensure that user insert correct information into the system.
Manage User	• The system should allow administrator to read and update the user information.
Manage Book	• The system should allow administrator to create, read, update and delete the book information. • The system should allow administrator to search for the required book.
Manage Box	• The system should allow administrator to create, read, update and delete box information. • The system show allow administrator to view the list of books in the selected box.
Manage Book Borrow	• The system should allow administrator and library prefect to carry out borrow and return book operation. • The system should allow administrator and library prefect to check the availability of the book.
Manage Box Borrow	• The system should allow administrator and library prefect to carry out borrow and return box operation. • The system should allow administrator and library prefect to check the availability of the box.
Manage Fine	• The system should allow administrator to penalize user whom exceed the dateline. • The system should allow administrator to accept payment made by user.
Manage Review	• The system should allow administrator to read views made by students and library prefects. • The system should allow students and library prefects to create, update and read review of the borrowed items
Reporting	• The system should auto generate report for administrator.
Check Borrowing	• The system should allow student and library prefects to check their borrowing history • Notify students of their eligibility for a lucky draw for every 10 books borrowed and returned.
Check Fine	• The system should allow student and library prefects to check their fine history
Chatbot	• The system should be able to answer basic inquiry from students and library prefect.
Manage Wishlist	• The system should allow students and library prefects to add, view and delete book or box from their wishlist
Manage Quote	• The system shall allow administrator to view quotes made by students and library prefects • The system should allow students and library prefects to create, read, update and delete quotes

Non-functional requirements in software engineering refer to specifications that detail how the software operates rather than its specific functionalities. Table 4 shows the non-functional requirements for the proposed system.

Table 4 *Non-Functional Requirements*

Module	Description
Availability	• The system should be accessible 24/7.
Performance	• The system should efficiently handle user loads during peak hours. • Each page must load within a time frame of 2 to 10 seconds. • The system's response time to user requests should not exceed 1 minutes. • The system should respond promptly and appropriately based on user requests.
Usability	• The system interface should be intuitive and user-friendly for intended users.
Security	• The system should verify users solely through their login usernames and passwords for access. • The system should restrict access to each user's interface solely to the respective authenticated user.

Defining and documenting user needs is of utmost importance, as they serve as the basis for system design User requirements encompass precise definitions of the tasks that the system will aid and the accompanying functionalities it will provide to enable these tasks. Table 5 displays the user requirements for the proposed system.

Table 5 *User Requirements*

No	User Requirements
1.	All users must have the capability to access the system by providing accurate username and password
2.	All users should be able to log out from the system.
3.	The administrator and library prefect shall be able to register new users into the system.
4.	The administrator should be able to read, and update user information.
5.	The administrator should be able to create, read, update and delete book information.
6.	The administrator should be able to create, read, update and delete box information.
7.	The administrator shall be able to see the list of books in the storage box.
8.	The administrator and library prefects shall be able to see the status availability of the boxes and books.
9.	The library prefects and administrator shall be able to manage the operation return and borrowing of boxes and books.
10.	The administrator shall be able to view fine, issue penalty and accept payments made by student and library prefects.
11.	The administrator shall be able to view the report.
12	The students and library prefects shall be able to create, update and read the reviews.
13	The administrator must be able to read the reviews.
14	The students and library prefects shall be able to view their fine and borrow records.
15	The system shall ensure only administrator can register new administrator into the system.

3.2 Use Case Diagram

Use case diagrams display multiple use cases originating from a single application, with actors depicted as tied to the use cases in which they participate. These diagrams illustrate the essential operations perceived by the user and the specific associations between these operations and other elements within the system's environment. Figure 2 is the use case diagram for the proposed system.

There will be a total of fifteen use cases, including login, register, manage user, manage book, manage box, manage book borrow, manage box borrow, manage fine, manage review, reporting, check borrowing, check fine, engage chatbot, manage quote and manage wishlist. Administrator, the primary actor of this system, will be able to access eleven modules, which are login, registration, manage user, manage book, manage box, manage book borrow, manage box borrow, manage fine, manage review, manage quote and reporting. Library prefects can access up to ten modules, whereas students can only access seven. Both students and library prefects will exclusively access check view, manage wishlist, engage chatbot and check fine.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

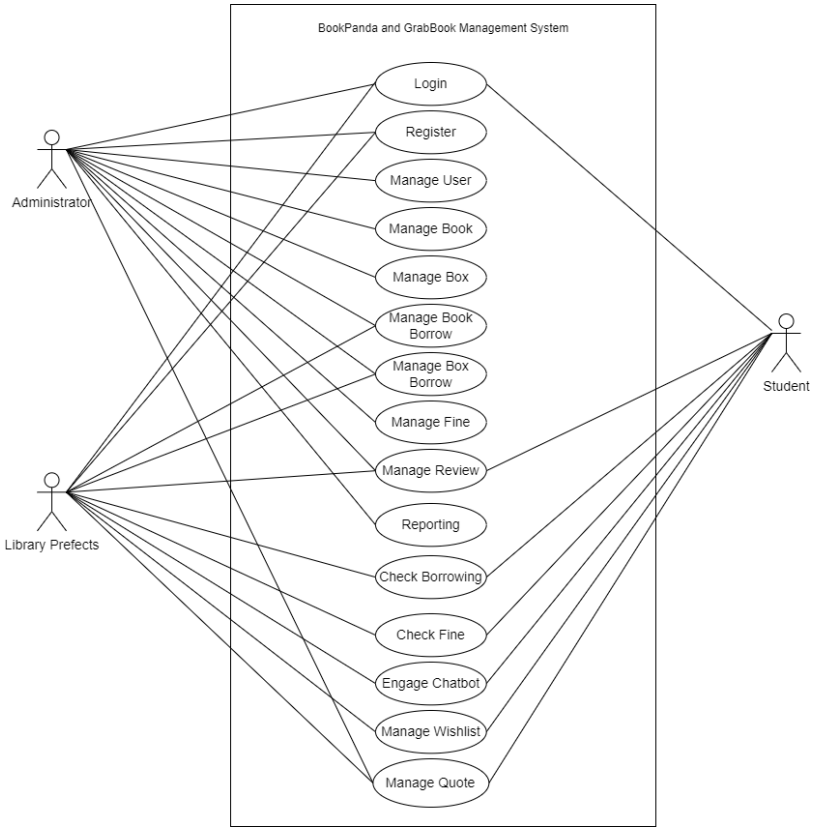

Fig. 2 *Use Case Diagram*

3.3 Class Diagram

The class diagram is a prominent diagram used in object-oriented systems. The purpose of this diagram is to illustrate the correlation between the entities and elucidate their functions and the services they offer [14]. There is a total of seventeen class in the class diagram with three classes which are administrator, library prefects and students are derived from user class through inheritance. Administrator class have a wide access of classes such as box, fine, BoxFines, books, book issue, box issue, review, quotes and reviewBox. In contrast library prefects can only access up to eight classes whereas students up to six classes. Each class is associated with a specific data type, which is states in its class attributes. The functions that is available for each class especially user classes are stated in the class operation. Figure 3 in Appendix A shows the class diagram for the proposed system.

3.4 Relational Database Schema

A relational database schema encompasses a set of table definitions, comprising of both stored base tables and derived views, together with constraints and derivation rules. A table scheme is a defined set of attributes (columns) that derive their values from domains. The schema contains a total of 15 tables, each with its own unique set of attributes.

i. User (userID, fullname, IC, email, birthdate, memberID, username, password, role, class, status, picture, clarify, reset_token_hash,reset_token_expired_at)
ii. Books (bookID, ISBN, author1, author2, title, publishDate, publicPlace, copy, pageNum, dateReceived, price, book_acquisition, genre, picture)
iii. Boxs (BoxSerialNum, category, dateCreate, bookQuantity, color, status, Boxpicture)
iv. IssueBook (issueBookID, memberID, bookID, IssueDate, returnDate, dueDate)
v. IssueBox (issueBoxID, memberID, BoxSerialNum, borrowDate, returnDate, dueDate)
vi. Fine (fineBookID, memberID, issueBookID, amount, isPaid, datePaid, fineID)
vii. BoxFines (boxFineID, memberID, issueBoxID, amount, isPaid, datePaid, fineID)
viii. FineBook (fineID, type, amount)
ix. FineBox(fineID, type, amount)
x. Review (reviewID, memberID, issueBookID, rating, DateReview, DateReviewEdit, review, isReview)

xi. ReviewBox (reviewBoxID, memberID, issueBoxID, rating, DateReview, DateReviewEdit, review, isReview)
xii. Book_distribution (book_distributionID, ISBN, BoxSerialNum, CopyCount)
xiii. Wishlist (WishlistID, memberID, ISBN, BoxSerialNum, Added_at)
xiv. Quote (quoteID, memberID, quote, tags,likes, Added_at)

3.5 User Interface Design

The effectiveness of a user interface lies in its ability to convey information clearly and concisely, employing the user's preferred language. Figure 4 shows the login interface for all the user involved which are administrator, library prefects and students. The system will validate the username and password entered by the user. Figure 5 is the manage user interface which is exclusively for administrator. Administrator can edit and view existing user account. Figure 6 shows the manage box interface for the proposed system. In this interface administrator can create, edit, read and delete boxes information.

Fig. 4 *Login*

Fig. 5 *Manage User*

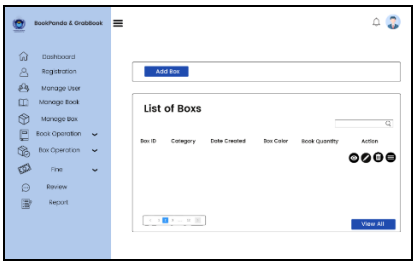

Fig. 6 *Manage Box*

Figure 7 display the manage book borrow module, which is the issue book interface. In this interface administrator will search for user ID, enter Book ISBN and click issue. Figure 8 is the, manage fine interface. Administrator can view students' fine details, accept payment and issue fines to students. Figure 9 display the manage review interface where administrator can view review made by users.

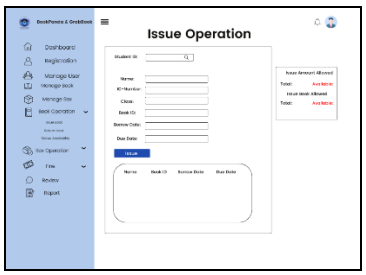

Fig. 7 *Issue Book*

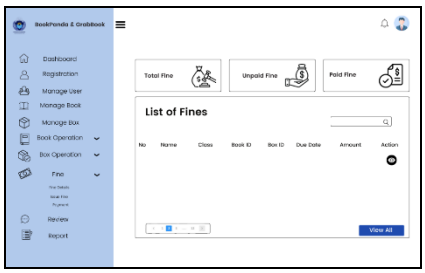

Fig. 8 *Manage Fine*

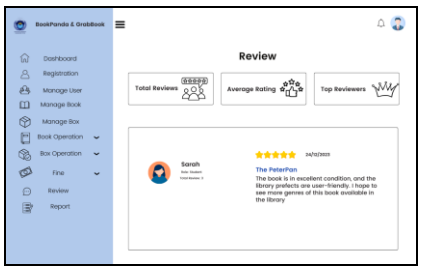

Fig. 9 *Manage Review*

Figure 10 display the reporting module interface. Administrator must select the preferred category to view the report. Check fine module interface is display in Figure 11 where students can check for their fine records. The difference between library prefects and students check fine interface is the features at the top navigation bar. Figure 12 display the check issue interface for library prefect. This interface allows library prefects to check for their issue details.

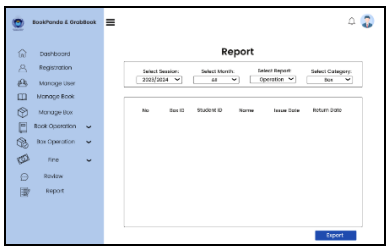

Fig. 10 *Report*

Fig. 11 *Check Fine*

Fig. 12 *Check issue*

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

4. Result and Discussion

This section will discuss two main parts, which are the implementation phase and the testing phase. The developed interface and code segments will be displayed in the implementation phase. The testing phase will be divided into two sections which are system testing and user acceptance testing.

4.1 Implementation Phase

The BookPanda and GrabBook Management System have been developed using Visual Studio Code, primarily utilizing frameworks like PHP, JavaScript, CSS, and HTML. The Google Books API is used to retrieve data from book ISBN. The recommendation system was developed using Python, employing the Cosine Similarity [15] algorithm. An unsupervised machine-learning technique that evaluates book attributes to generate choices. This recommendation modules integrates the Flask framework, SQLAlchemy for the database, and pandas for data manipulation. The Cosine Similarity from the sci-kit-learn library allows for calculating similarities across books, enhancing the system's ability to provide accurate suggestions. The chatbot has been developed using Natural Language Processing (NLP) with Wit.ai [16] and trained to understand utterances for each intent.

Figure 13 shows the login page for all users in the system. After logging in successfully, each user is directed to their respective page. Figure 14 presents the registration form interface for administrators, who must complete all required fields. Upon successful registration, a success message appears. If there is any incorrect information, an error message is displayed.

Fig. 13 *Login*

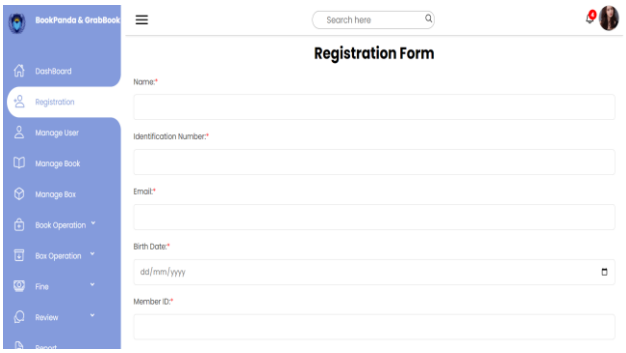

Fig. 14 *Registration*

Figure 15 shows the book management interface for administrators, allowing them to create, read, update, and delete books, which are displayed in an alphabetically ascending table. It includes a search and sort function for genres, latest, oldest, and all books, with a confirmation message for deletions. Figure 16 displays the book borrowing management interface, where administrators can enter or scan a member ID to retrieve the member's name, identification number, and class.

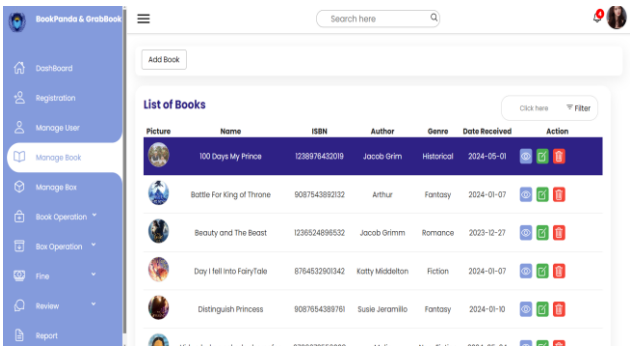

Fig. 15 *Manage Book*

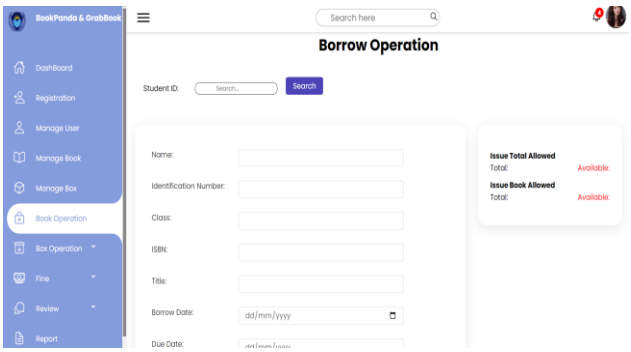

Fig. 16 *Manage Book Borrow*

Figure 17 depicts the user interface used to oversee the specific information related to book fines. The table provides a comprehensive breakdown of information for both students and library prefects. Administrators can sort the table based on several criteria, such as payment status (paid or unpaid) and other categories of fines, including late returns, missing items, and damages. Figure 18 illustrates the administrator dashboard. The

dashboard showcases box and book issue records in a tabular style, accompanied by a histogram illustrating monthly issue records. The pie chart also displays the status of book borrowing.

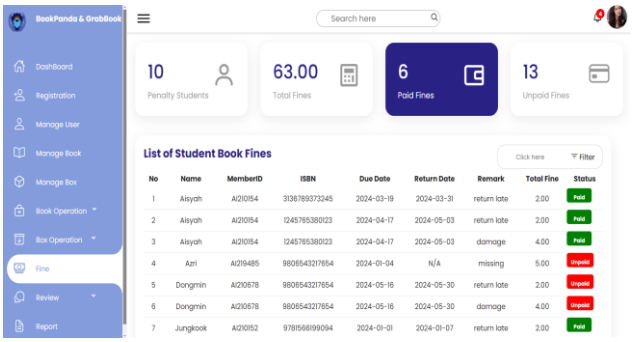

Fig. 17*Manage Fine Details*

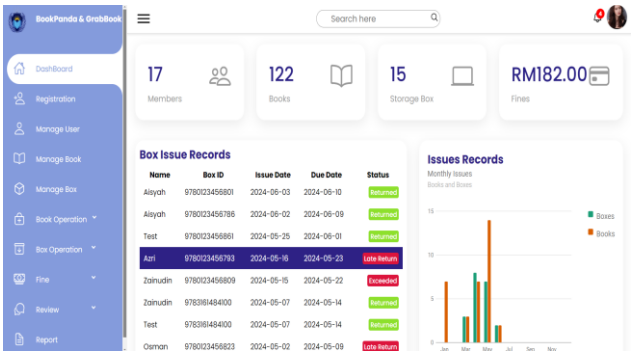

Fig. 18 *Dashboard*

Figure 19 shows the student dashboard. At the bottom, there is a flashcard streak displaying the student's active achievements. Notifications inform students about overdue borrowed items and recently added books or boxes. Figure 20 shows the interface for managing book reviews. Students can add, edit, and view reviews they have made.

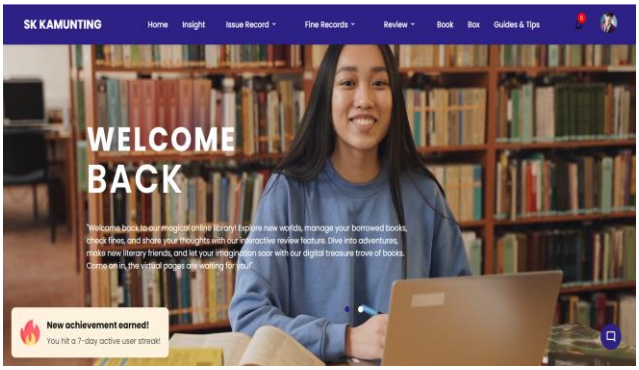

Fig. 19 *Student Dashboard*

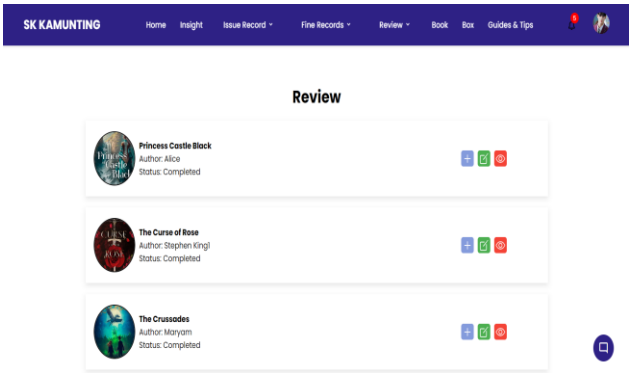

Fig. 20 *Book Review*

Figure 21 shows the list of boxes available in the library. Clicking on a box will display its reviews and details. Students and library prefects can also add boxes to their Wishlist. Figure 22 presents the interface for checking student fines. In this module, students can view a list of fines for books, the amounts due, and their status as paid or unpaid.

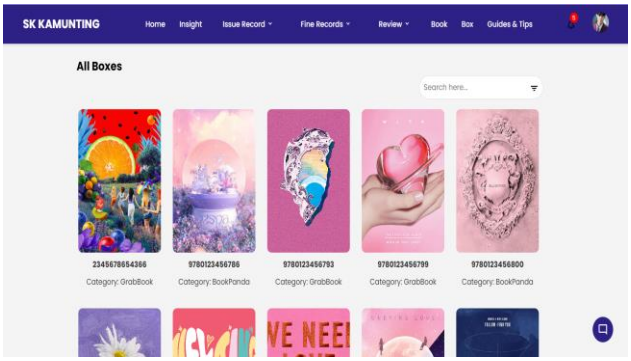

Fig. 21 *List of Box*

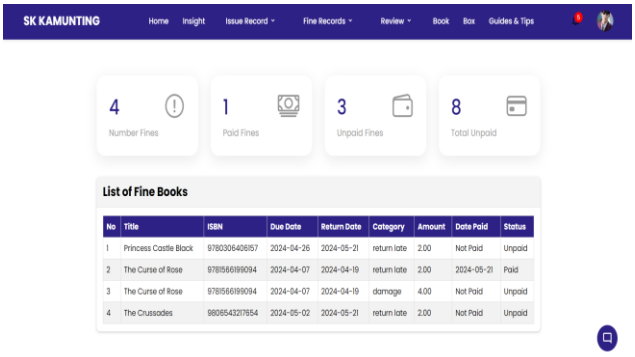

Fig. 22 *Check Fine*

Figure 23 display the book recommendation that will be displayed to towards respective students based on the implemented cosine similarity algorithm. Figure 24 shows, the chatbot interface for students. Students are able to ask basic inquiries regarding book, box and library history. Upon clicking the chat icon, the chatbot box will popup.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

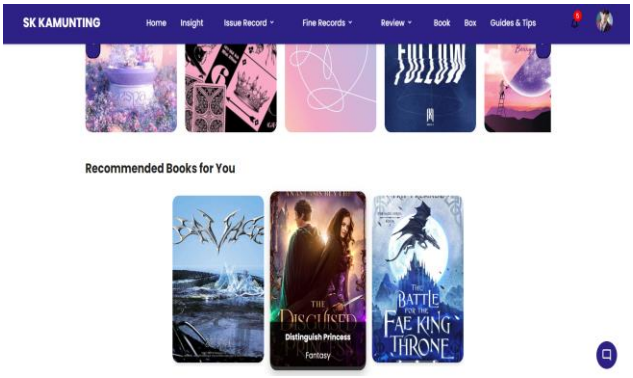

Fig. 23 *Book Recommendation*

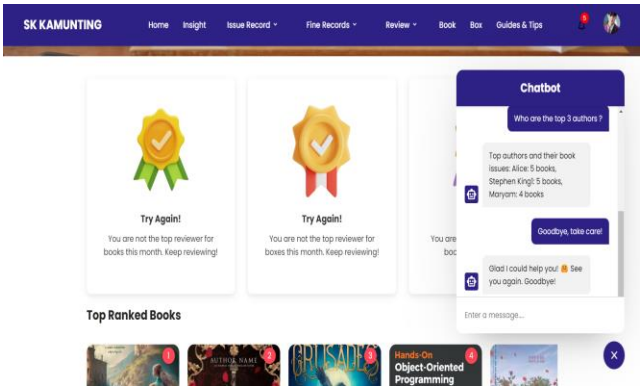

Fig. 24 *Chatbot*

Figure 25 shows the manage report interface for administrators. Administrators must select the session, month, report type, and category before submitting the report. Additionally, they have the option to print the report in PDF format. Figure 26 displays the Wishlist interface for library prefects. Library prefects can view and delete books from their Wishlist using this interface.

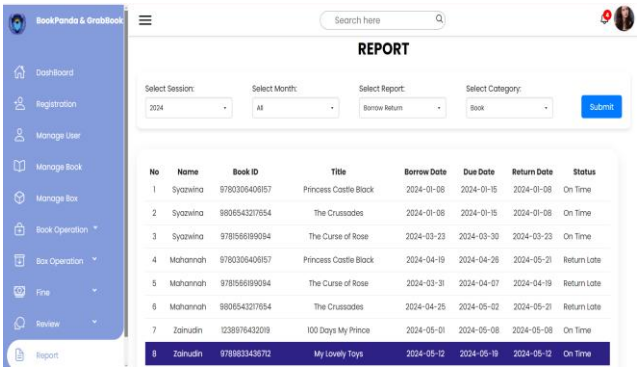

Fig. 25 *Manage Report*

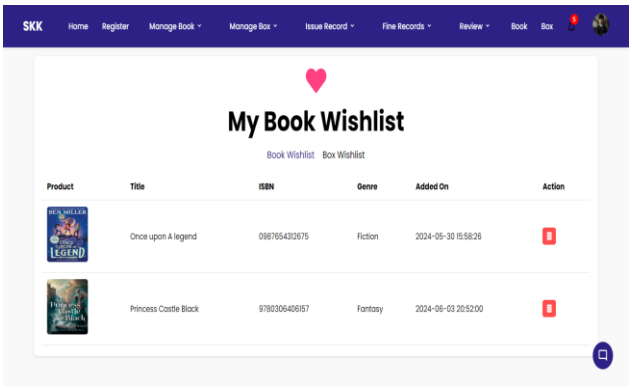

Fig. 26 *Wishlist*

Figure 27 display the issuebook code segments. The `issueBook` function manages the process of book issuing by verifying the validity of form fields and verifying the availability of the book as well as the borrowing limitations of the member. The program collects the data entered in the form, transmits it to `issueBook1.php` via a POST request, and handles the response from the server to display either a popup indicating success or an error message. If the book is not accessible or if the necessary information is not provided, suitable alerts are displayed.

```
async function issueBook(event) {
    event.preventDefault();

    var bookAvailabilityMsg = document.getElementById('book_availability_msg').textContent.trim();
    if (bookAvailabilityMsg === 'Book not available') {
        alert('Sorry, this book is fully borrowed and currently unavailable.');
        return;
    }

    var memberID = document.querySelector('input[name="memberID"]').value.trim();
    var isbn = document.getElementById('ISBN').value.trim();
    var borrowDate = document.getElementById('borrowD').value.trim();
    var dueDate = document.getElementById('dueD').value.trim();

    if (!memberID || !isbn || !borrowDate || !dueDate) {
        alert('Please fill in all required fields: Member ID, ISBN, Borrow Date, and Due Date.');
        return;
    }

    const canBorrow = await checkBorrowLimit(memberID);
    if (!canBorrow) return;

    var formData = new FormData();
    formData.append('memberID', memberID);
    formData.append('ISBN', isbn);
    formData.append('borrowDate', borrowDate);
    formData.append('dueDate', dueDate);

    fetch('issueBook1.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.text())
    .then(text => {
```

Fig. 27 *IssueBook Code Segment*

The given code in figure 28 generates a Flask web application that utilizes the Cosine Similarity algorithm to suggest books to users based on their borrowing history. The system establishes a connection with a MySQL database in order to retrieve information about books and borrowing records. It then combines these datasets to identify books that the user has not borrowed. It translates genres into a binary matrix to calculate similarities. Most comparable books are suggested. Figure 29 shows the chatbot code segments that utilize javascript. The handleIntent function analyzes the Wit.ai answer to determine the user's intention and entities. If no intention is detected, an error message is displayed. It can process simple responses, precise inquiries about books or boxes, or generic inquiries for recognized purposes.

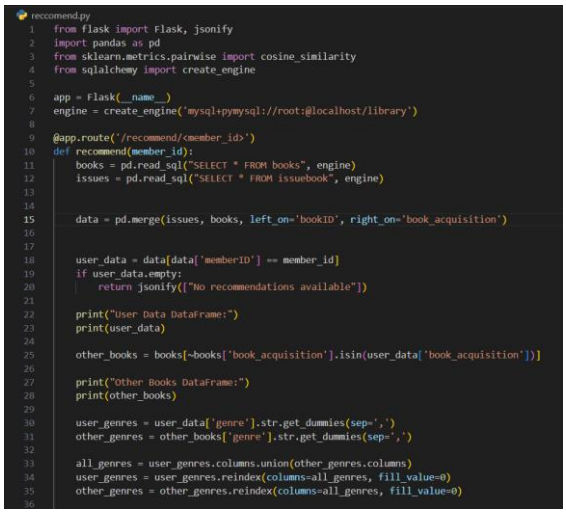

Fig. 28 *Python Code Segment*

Fig. 29 *Chatbot Code Segment*

Figure 30 displays a sample of the data used to train the chatbot in Wit.AI. Hundreds of inquiries were generated to educate the chatbot, emphasizing distinct elements to trace each purpose. The chatbot was effectively trained using less than 30 categories in total.

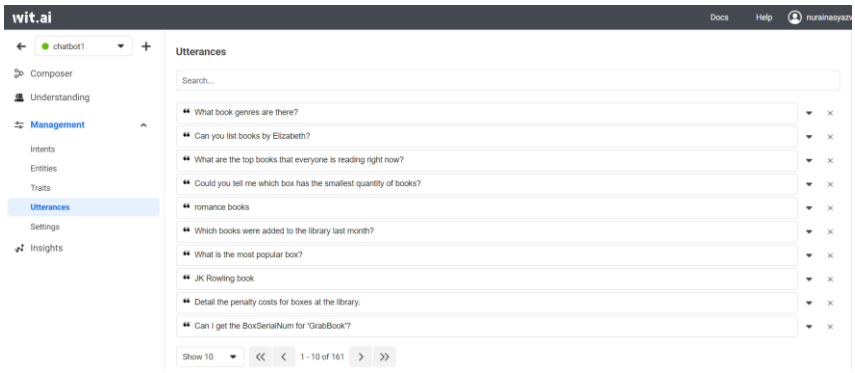

Fig. 30 *Wit.Ai Sample Data*

4.2 Testing

The testing phase will commence upon the system's complete development. Testing is a critical phase that necessitates execution after implementation. System testing aims to confirm that the module's critical and secondary functionalities operate effectively. Conversely, user acceptability testing will be implemented utilizing a Google Form distributed to users. The response will facilitate feedback collection to verify that the system aligns with their expectations and requirements. Testing helps to see whether the system can achieve its intended purpose by and it functions efficiently

4.2.1 System Testing

The 15 modules will be tested to determine whether they function as intended. The Requirement Traceability Matrix (RTM) is a document that assists in illustrating the connection between requirements and other artefacts, guaranteeing that all requirements are considered during the testing phase. This procedure aims to verify that each module functions properly and satisfies the specified specifications. Table 6 displays the test case list for the developed system.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

Table 6 *Test Case List*

Software Requirement Specification	Test Cases ID	Description	Test Result (Pass/Fail)
TEST_100(Login Module)			
FR_REQ_101	Test_100_001	User able to login using the correct username and password	Pass
FR_REQ_102	Test_100_002	User able to prompt another login for invalid credentials	Pass
FR_REQ_103, CR_REQ_106	Test_100_003	An error message is display upon unsuccessful login credentials	Pass
FR_REQ_104	Test_100_004	User will be directed to respective page based on role upon login	Pass
FR_REQ_105	Test_100_005	User can re-enter password using forgot password	Pass
CR_REQ_107, QR_REQ_108	Test_100_006	System can handle exceptions	Pass
TEST_200 (Registration Module)			
FR_REQ_201, FR_REQ_202	Test_200_001	Administrator and library prefect can register new user into the system	Pass
FR_REQ_203	Test_200_002	A required message is display if user fails to complete all the mandatory information	Pass
FR_REQ_204, CR_REQ_206, QR_REQ_208	Test_200_003	An error message will be display if user already exist	Pass
CR_REQ_205, CR_REQ_207	Test_200_004	Administrator role can only be register by administrator	Pass
TEST_300 (Manage User Module)			
FR_REQ_301	Test_300_001	Administrator can view user information	Pass
FR_REQ_302	Test_300_002	Administrator can edit user information	Pass
CR_REQ_303	Test_300_003	Only administrator can access this module	Pass
QR_REQ_304	Test_300_004	System updates the data based on the latest administrator input	Pass
QR_REQ_305	Test_300_005	System constraint administrator from editing mandatory information like identification number and password	Pass
TEST_400 (Manage Book Module)			
FR_REQ_401	Test_400_001	Administrator can search for respective book	Pass
FR_REQ_402	Test_400_002	Administrator can view book information	Pass
FR_REQ_403	Test_400_003	Administrator can add book information	Pass
FR_REQ_404	Test_400_004	Administrator can edit book information	Pass
FR_REQ_405	Test_400_005	Administrator can delete book information	Pass
FR_REQ_406	Test_400_006	A 'required field' message is displayed if any mandatory field is left unfilled	Pass
FR_REQ_407	Test_400_007	System displays error message if administrator attempt to add information that does not meet specified validation criteria	Pass
FR_REQ_408	Test_400_008	System displays success message for success edits, add and delete book	Pass
CR_REQ_409	Test_400_009	System will only delete the book after confirmation	Pass
CR_REQ_410	Test_400_010	System only allows the book to be added after filling up all mandatory information with correct validation	Pass
QR_REQ_411	Test_400_011	Only administrator can access this module	Pass
QR_REQ_412	Test_400_012	All the database is updated based on the create, update and delete function to ensure data integrity	Pass
TEST_500 (Manage Box Module)			
FR_REQ_501	Test_500_001	Administrator can search for the required box	Pass

Table 6: (*cont)*

FR_REQ_502	Test_500_002	Administrator can view box information	Pass
FR_REQ_503	Test_500_003	Administrator can add box information	Pass
FR_REQ_504	Test_500_004	Administrator can edit box information	Pass
FR_REQ_505	Test_500_005	Administrator can delete box information	Pass
FR_REQ_506	Test_500_006	Administrator can see the list of books in box	Pass
FR_REQ_507	Test_500_007	System displays error message for invalid format	Pass
FR_REQ_508	Test_500_008	System able to display success message upon success add, edit and delete box	Pass
CR_REQ_509	Test_500_009	Box can only be added after administrator filled up all mandatory information with correct format	Pass
CR_REQ_510	Test_500_010	Administrator can only delete the box after confirmation	Pass
CR_REQ_511	Test_500_011	Only administrator can access this module	Pass
QR_REQ_512	Test_500_012	Database is updated based on the create, update and delete operations to ensure data integrity	Pass
TEST_600 (Manage Book Borrow Module)			
FR_REQ_601	Test_600_001	System allow book to be issue to library prefects and students	Pass
FR_REQ_602	Test_600_002	System allows return book made by students and library prefects	Pass
FR_REQ_603	Test_600_003	System allows administrator and library prefects to see the book availability status	Pass
FR_REQ_604	Test_600_004	System displays error message if the book is not found or already return	Pass
CR_REQ_605	Test_600_005	System displays alert message if mandatory field remain unfilled	Pass
CR_REQ_606	Test_600_006	System displays alert message if students and library exceed limit issue	Pass
CR_REQ_607	Test_600_007	System prohibits students from using this module	Pass
QR_REQ_608	Test_600_008	System able to retrieved data upon user request	Pass
TEST_700 (Manage Box Borrow Module)			
FR_REQ_701	Test_700_001	System allows students and library prefect to issue box	Pass
FR_REQ_702	Test_700_002	System allows students and library prefects to return box	Pass
FR_REQ_703	Test_700_003	Administrator and library prefects can see the status availability of the box	Pass
FR_REQ_704	Test_700_004	System can show alert message when a mandatory field remain unfilled	Pass
CR_REQ_705	Test_700_005	Students are prohibiting from managing this module	Pass
CR_REQ_706	Test_700_006	System shows alert message if students and library prefects exceed issue limits	Pass
QR_REQ_707	Test_700_007	System capable of retrieving data upon user request	Pass
TEST_800 (Manage Fine Module)			
FR_REQ_801	Test_800_001	Administrator able to view fine details	Pass
FR_REQ_802	Test_800_002	Administrator can issue penalties based on the rules that user violates	Pass
FR_REQ_803	Test_800_003	Administrator can submit payment made by user	Pass
CR_REQ_804	Test_800_004	An alert message will be shown upon absence of fine	Pass
CR_REQ_805	Test_800_005	Only administrator able to access this module	Pass
QR_REQ_806	Test_800_006	Administrator receive success alert message upon success issue fine and payment fine	Pass
TEST_900 (Manage Review Module)			
FR_REQ_901	Test_900_001	Administrator able to view reviews made by user	Pass
FR_REQ_902	Test_900_002	Students and library prefects able to add review of borrowed items	Pass
FR_REQ_903	Test_900_003	Students and library prefects able to edit review of borrowed items	Pass
FR_REQ_904	Test_900_004	Students and library prefects able to view review	Pass

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

Table 6: (*cont)*

FR_REQ_905	Test_900_005	System shows alert message if student/library prefects attempt to add a new review instead of editing existing one	Pass
CR_REQ_906	Test_900_006	Review will only be submitted after library prefects/ students enter the rate and review.	Pass
CR_REQ_907	Test_900_007	System shows an alert message if library prefects/students try to submit review without filling up all information's	Pass
QR_REQ_908	Test_900_008	System able to show success message for every edit and add review operations	Pass
TEST_1000 (Reporting Module)			
FR_REQ_1001	Test_1000_001	Administrator can choose report category	Pass
FR_REQ_1002	Test_1000_002	System able to show the report based on user report selection	Pass
FR_REQ_1003	Test_1000_003	Administrator able to print report in pdf	Pass
CR_REQ_1004	Test_1000_004	Module can only be accessible to administrator	Pass
QR_REQ_1005	Test_1000_005	System able to fetch data less than 5 seconds after administrator initiate the request	Pass
TEST_1100 (Check Borrow Module)			
FR_REQ_1101	Test_1100_001	Library prefects/ students can view borrow records based on their memberID	Pass
FR_REQ_1102	Test_1100_002	The system displays student eligibility for a lucky draw after every 10 books borrowed and returned.	Pass
CR_REQ_1103	Test_1100_003	System prohibits library prefects/student from editing the borrow information	Pass
QR_REQ_1104	Test_1100_004	System able to fetch data in less than 5 seconds	Pass
TEST_1200 (Check Fine Module)			
FR_REQ_1201	Test_1200_001	Library prefects/ students able to check their own fine records	Pass
CR_REQ_1202	Test_1200_002	System prohibits library prefects/students to edit the fine records	Pass
QR_REQ_1203	Test_1200_003	System able to fetch the fine records in less than 5 seconds	Pass
TEST_1300 (Engage Chatbot Module)			
FR_REQ_1301	Test_1300_001	Chatbot able to answer basic questions	Pass
CR_REQ_1302	Test_1300_002	Module is only available to students and library prefects	Pass
QR_REQ_1303	Test_1300_003	Chatbot able to answer questions in less than 1 minutes	Pass
TEST_1400 (Manage Wishlist Module)			
FR_REQ_1401	Test_1400_001	Students/Library prefects able to add book or box to their wishlist	Pass
FR_REQ_1402	Test_1400_002	Students/Library prefects able to view their wishlist book or box	Pass
FR_REQ_1403	Test_1400_003	Students/Library prefects able to delete book or box from their wishlist	Pass
CR_REQ_1404	Test_1400_004	Administrator prohibit from accessing this module	Pass
QR_REQ_1405	Test_1400_005	Students/ library prefects can perform actions without any disruption	Pass
TEST_1500 (Manage Quote Module)			
FR_REQ_1501	Test_1500_001	Administrator able to view quotes	Pass
FR_REQ_1502	Test_1500_002	Library prefects/ students can add quotes	Pass
FR_REQ_1503	Test_1500_003	Library prefects/ students can edit quotes	Pass
FR_REQ_1504	Test_1500_004	Library prefects/ students can view quotes	Pass
FR_REQ_1505	Test_1500_005	Library prefects/ students can delete quotes	Pass

Table 6: (*cont)*

CR_REQ_1506	Test_1500_006	Library prefects/ students cannot submit more than 2 quotes per day	Pass
QR_REQ_1507	Test_1500_007	System able to show success message for every edit, delete and add review operations	Pass

The overall test case results are summarized in a tabular format in Table 4.2, which clearly and concisely represents the testing outcomes.

Table 7 *Overall Test Case Result*

Test Case ID	Total Test Cases	Total Passed
TEST_100	8	8
TEST_200	8	8
TEST_300	5	5
TEST_400	12	12
TEST_500	12	12
TEST_600	8	8
TEST_700	7	7
TEST_800	6	6
TEST_900	8	8
TEST_1000	5	5
TEST_1100	4	4
TEST_1200	3	3
TEST_1300	3	3
TEST_1400	5	5
TEST_1500	7	7
Total	101	101

4.2.2 User Acceptance Testing

User acceptance testing entails gathering feedback from the intended end-users. Multiple factors of the system will undergo testing, encompassing its functioning, interface, and usability. A meeting was conducted with the stakeholder to collect her feedback The stakeholder will conduct system testing in their role as administrators, while a total of 19 participants, including students and library prefects, will be involved. The total number of responses, including the stakeholders, will be 20. Questions 1 to 9 will be rated on a scale of 1 to 5, while question 10 will require a written response.

The questions related to system interface testing are shown in Figure 31. According to the data, most users are happy with the system's UI. Following a meeting with the stakeholder, she expressed her satisfaction with the interface because it keeps the former theme, and the administrator interface is easy to use and navigate because it roughly mimics the structure of the old system.

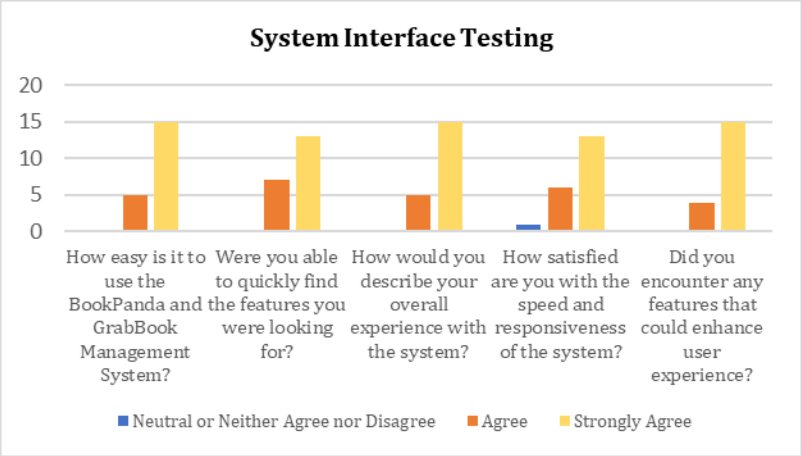

Fig. 31 *System Interface Testing Result*

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

Figure 32 illustrates the results of the system's functionalities. The stakeholder expressed great satisfaction, emphasizing that her requirements had been fulfilled. She showed great enthusiasm for the new modules, such as the chatbot, review, wishlist, achievement, and quotation functions. Fourteen participants expressed a strong agreement, while six people expressed a general agreement that all features operated as expected. Eighteen participants expressed strong agreement, while two agreed on their willingness to promote the system to others, which suggests its effectiveness and user-friendliness. In addition, fifteen respondents strongly agreed, while five agreed on the benefits of the new duties for students and library prefects. Sixteen individuals strongly agreed, while four participants agreed about the system's fulfilment of their expectations.

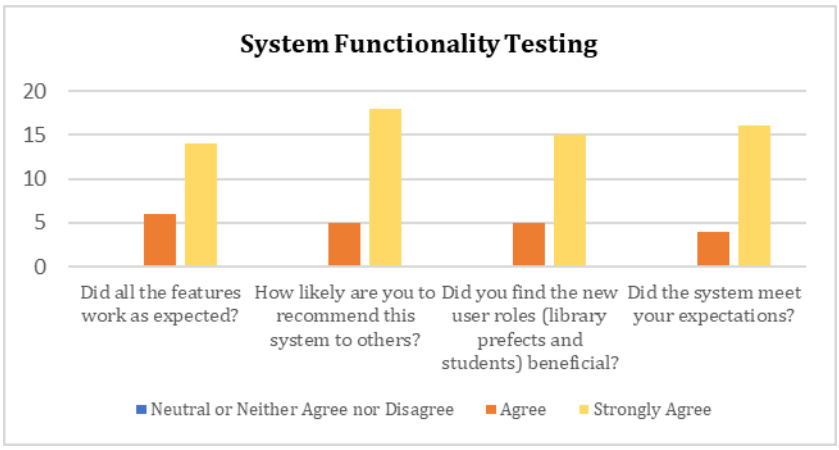

Fig. 32 *System Functionality Testing Result*

Figure 33 displays the responses regarding bug experiences. Most users reported that they did not encounter any errors while navigating the system, indicating a smooth and error-free user experience.

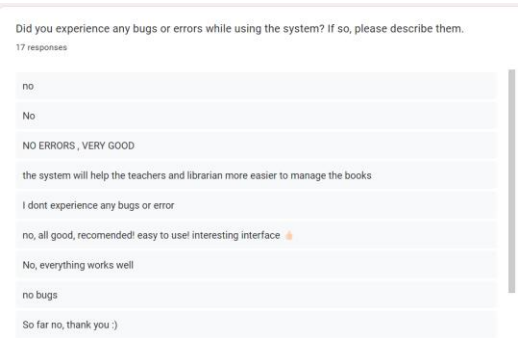

Fig .33 *System Functionality Testing Result*

Overall, the stakeholder, Cikgu Aishah, is satisfied with the outcome of the developed system, indicating that all system requirements and project objectives have been successfully achieved.

5. Conclusion

In conclusion, the development of the BookPanda and GrabBook Management System has effectively addressed critical challenges in the SK Kamunting library system by engaging administrators, library prefects, and users with its unique modules. The system supports multiple user roles, securely manages data, facilitates reviews, offers a responsive chatbot, and provides personalized book recommendations, significantly enhancing the efficiency of library management. However, limitations such as non-responsiveness to mobile devices, reliance on internet connectivity, restricted chatbot inquiries, and a basic recommendation system persist. Therefore, to drive future advancements, integrating web and mobile applications is strongly recommended to broaden accessibility. Enhancing the chatbot to handle a broader range of inquiries and upgrading the recommendation system to employ supervised machine learning would improve user experience. Additionally, incorporating a dashboard for library events would keep users informed and engaged. This strategic evolution promises to extend the system's reach and usability, catering to diverse user needs while fostering further innovation in library management systems. Continued research and development could significantly impact educational institutions by refining library systems to meet evolving technological and user-related demands.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] Moran, B. B., & Morner, C. J. (2017). Library and information center management. Bloomsbury Publishing USA.
[2] Buckingham, R.A., Hirschheim, R., Land, F.F., & Tully, C. (1987). Information Systems Education: Recommendations and Implementation.
[3] Teixeira, L., Ferreira, C., & Santos, B.S. (2008). Web-Enabled System Design for Managing Clinical Information.
[4] Felke-Morris, T. (2020). Web Development and Design Foundations with HTML5, Global Edition.
[5] R. A. Muzaki, O. C. Briliyant, M. A. Hasditama and H. Ritchi. (2020) "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. 2020 International Workshop on Big Data and Information Security (IWBIS)(pp85-90). Depok, Indonesia, 2020: IEEE
[6] O'Kane, M. (2022). A web-based introduction to programming: Essential algorithms, syntax, and control structures using PHP, HTML, and mariadb/mysql. Carolina Academic Press.
[7] Green, T. J., & Dias, T. (2010). Foundation flash CS5 for designers (1st ed.). Tom Green.
[8] Crockford, D. (2008). JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.".
[9] DuBois, P. (2013). MySQL (5th ed). Addison-Wesley.
[10] Sharma, S., Mishra, S., Gupta, S., & Kumar, S. (2022). Library Management System. International Journal for Research in Applied Science and Engineering Technology.
[11] Dennis, A., Wixom, B. H., and Roth, R. M. 2003. Systems Analysis and Design. 3rd Edition, John Wiley & Sons Inc.
[12] Sabastian, R., Arun, P., Krishna, N. G., & Faris, R. M. (2021). International Research Journal of Modernization in Engineering Technology and Science. *International Research Journal of Modernization in Engineering Technology and Science*, *3*(04).
[13] Shah, T., & Patel, S. V. (2016). A novel approach for specifying functional and non-functional requirements using rds (requirement description schema). Procedia computer science, 79, 852-860.
[14] Bridgeland, D. M., & Zahavi, R. (2008). Business modeling: a practical guide to realizing business value. Morgan Kaufmann.
[15] Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information sciences*, *307*, 39-52.
[16] Qaffas, A. A. (2019). Improvement of Chatbots semantics using wit. ai and word sequence kernel: Education Chatbot as a case study. *International journal of modern education and computer science*, *11*(3), 16.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1160-1179

Appendix A: Class Diagram

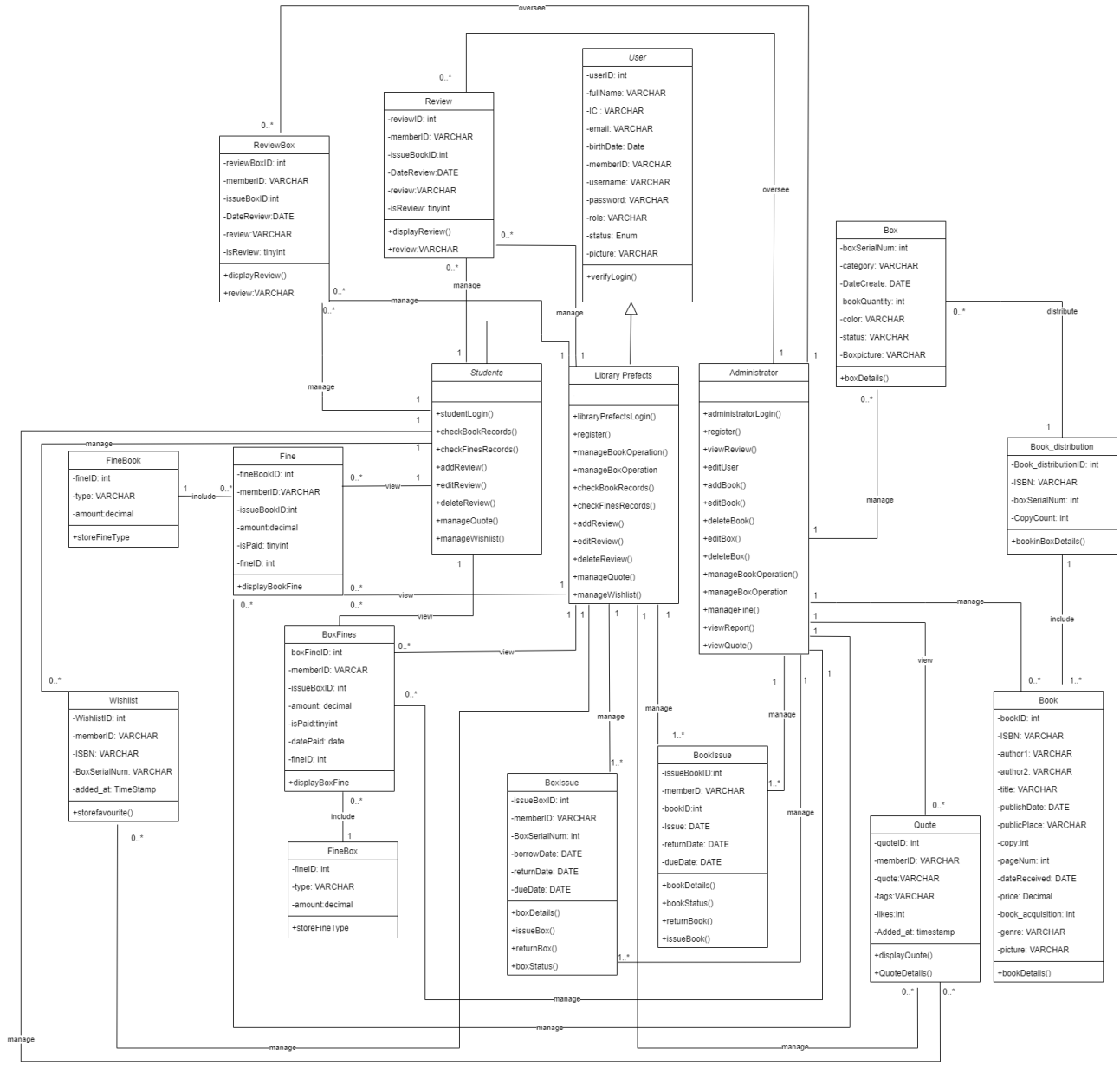

Fig. 3 *Class Diagram*