



FINAL REPORT

Book-Nest

Submitted By

Rabia BiBi	2021-Gcuf071099
Hira Fatima	2021-Gcuf-071095
Iqra Shahbaz	2021-Gcuf -056596



Department of Computer Science
Govt. Graduate College Samundri, Faisalabad

2024

CERTIFICATE

This is to certify that **Rabia BiBi , Hira Fatima, Iqra Shahbaz** bearing Registration No. 2021-Gcuf071099, 2021-Gcuf-071095, has completed the final project titled **“Book Nest”** at the **Department of Computer Science, Government Graduate College Samundri**, to fulfill the partial requirement for the degree of **BS - CS**.

Supervisor

M Sohaib Yousaf

Signature: _____

Internal Panel

Member 1: _____

Signature: _____

Project Coordinator

Head of Department

Principal

DECLARATION

This work reported in this final project was carried out by us under the supervision of **Mam Farhat Naaz**, Department of Computer Science, **Govt. Graduate College Samundri**. We hereby declare that the title of the project is **Book-Nest** and the contents of the project are the product of my research and no part has been copied from any published source. (Except the references, standard mathematical or genetic models /equations /formulas /protocols, etc.) We further declare that this work has not been submitted for the award of any other degree /diploma. The institution may take action if the information provided is found inaccurate at any stage.

Name:	Rabia BiBi
Registration No.	2021-Gcuf071099
Name:	Hira Fatima
Registration No.	2021-Gcuf-071095
Name:	Iqra Shahbaz
Registration No.	2021-Gcuf -056596

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to **Allah Almighty** for giving me the strength, knowledge, and perseverance to successfully complete my final year project, titled "**Book-Nest**". I extend my heartfelt thanks to my respected supervisor, **Mam Farhat Naaz**, for their valuable guidance, support, and constructive feedback throughout the development of this project. Their encouragement and mentorship played a vital role in the successful completion of this work. I am also thankful to all the **faculty members of the Computer Science Department**, whose dedication and efforts in teaching laid a strong foundation that enabled me to undertake and complete this project. My special thanks go to my friends and classmates for their cooperation, motivation, and continuous support during the development of this system.

ABSTRACT

BookNest is a web-based Library Book Management System designed to simplify and digitize the process of book handling in modern libraries. The system comprises two major modules: an **Admin Dashboard** and a **User Website**. Through the dashboard, administrators can efficiently manage the digital library by adding, editing, deleting book records, and uploading PDF versions of books. The public-facing website allows users to browse the complete collection of available books. To download any book, users must register and log in, ensuring secure access to digital content. This login mechanism ensures that only authenticated users can download books, maintaining data integrity and controlled access. The system is developed using modern web technologies and offers an intuitive user interface for both admins and users. **Book Nest** not only reduces the manual effort involved in managing a library but also provides users with a convenient and secure way to access digital resources anytime, anywhere. This project aims to promote digital literacy, improve accessibility to educational resources, and bring traditional library systems into the digital age.

TABLE OF CONTENTS

CHAPTER NO 1 INTRODUCTION	1
1.1 Purpose	1
1.2 Project Scope	1
1.3 Project Planning.....	2
1.4 Technology We Used	3
1.4.1 React Native Technology.....	3
1.4.2 Hardware Requirement React JS	3
1.4.3 Text Editor	4
1.4.4 Node.js	4
1.4.5 Google Chrome.....	5
1.5 Book Nest.....	6
1.5.1 Admin Dashboard	6
1.6 Modules.....	6
1.7 Benefits.....	7
1.7.1 Benefits for Consumers	7
1.8 Objective	8
1.9 Feature	8
1.9.1 Admin Feature	8
1.9.2 Website Feature	9
CHAPTER NO 2 BACKGROUND.....	10
2.1 Background Research.....	10
2.2 Existing Technology.....	10
2.3 Reason of Project	10
2.4 Tools	11
2.4.1 React	11
2.4.2 Node.js	12
2.4.3 MongoDB	13
2.5 Tool Used to Develop “Website”	14
2.5.1 Visual Studio Code	14
CHAPTER NO 3 SYSTEM REQUIREMENTS ANALYSIS	16
3.1 Introduction of the Requirement.....	16

3.2 Software Requirement Specification (SRSs)	16
3.3 Gathering requirements from client	16
3.4 Typical Requirements Gathering Process	17
3.5 Capture Better Requirements.....	17
3.6 Different Types of Software Requirements	18
3.6.1 Business Requirements (BR)	19
3.6.2 Market Requirements (MR)	19
3.6.3 Functional Requirements (FR) – Use Cases	19
3.6.4 Non-Functional Requirements (NFR).....	19
3.6.5 UI Requirements (UIR)	19
3.7 Requirements for Book Nest.....	20
3.7.1 Functional Requirements	20
3.8 Functional Requirement of BookNest Website:.....	20
3.8.1 Order Booking	20
3.8.2 Add/Remove/Edit Products from inventory	21
3.8.3 Add categories	21
3.9 Non-Functional Requirement of Book NEST	21
3.9.1 Easy to understand Interface for any non-technical user	22
3.9.2 Reliability	22
3.9.3 Security	22
3.9.4 Usability.....	23
CHAPTER NO 4 SYSTEM DESIGN	24
4.1 Methodology	24
4.2 Unified Modeling Language.....	24
4.3 Designing Technique.....	25
4.4 Waterfall Model	25
4.4.1 Why we used waterfall model.....	25
4.4.2 Diagram of Waterfall Model.....	26
4.5 System Features	26
4.5.1 System Features 1	26
4.5.2 System Features 2	27
4.6 Use case Diagram	27
4.6.1 Use case 1:	28
4.6.2 Use case 2:	29

4.7 Entity Relation Diagram	30
4.7.1 The elements of an ERD are:	30
4.7.2 Steps involved in creating an ERD include:	31
4.8 System Sequence Diagram	31
4.8.1 Overview.....	31
4.8.2 Advantages of sequence diagram.....	31
4.8.3 Sequence Diagram	31
CHAPTER NO 5 TESTING & IMPLEMENTATION.....	33
5.1 Project Testing	33
5.2 Purposes of testing	33
5.3 Benefits of Automated Software Testing	33
5.3.1 Efficient Testing	34
5.3.2 Upgradation and Reusability.....	34
5.3.3 Consistency	34
5.3.4 Unique Programmability and Shelf Life	35
5.3.5 User Environment Simulation.....	35
5.4 The Basic Levels of Testing are	35
5.5 Users need acceptance Testing.....	36
5.5.1 Who performs user acceptance testing?	36
5.6 Prerequisites of User Acceptance Testing	36
5.7 System Testing.....	36
5.8 Analogy	36
5.8.2 System Test Plan.....	37
5.8.3 When is it performed?.....	37
5.8.4 Who performs it?	37
5.9 Unit Testing	37
5.9.1 Who performs it?	38
5.10 Integration Testing	38
5.11 System Testing.....	39
5.12 Acceptance Testing	39
5.13 Validation Check.....	39
5.14 Implementation	39
5.15 System Verification.....	40

5.16 Alpha Testing	41
5.17 Beta Testing	41
5.18 Types of Beta Testing	41
CHAPTER NO 6 USER MANUAL	47
6.1 Customer Panel	47
6.1.1 Main Screen	48
6.1.2 Books	49
6.1.3 Book Detail	50
6.2 Admin Panel	51
6.2.1 Dashboard	51
6.2.2 Books	51
6.2.3 Edit Book	52
6.2.4 Add Books	52
6.3 Code of Main Activity	53
6.3.1 Main Activity of React App.js file	53

LIST OF FIGURES

Figure 4.1: Waterfall Model	26
Figure 4.2: Use case of Admin	28
Figure 4.3: Use case of Admin	29
Figure 4.4: Entity Relation Diagram.....	30
Figure 4.5: System Sequence Diagram.....	32
Figure 5.1: Testing Diagram.....	30
Figure 6.1: Home Screen	48
Figure 6.4: Products	49
Figure 6.5: Product Detail.....	50
Figure 6.8: Categories.....	51
Figure 6.9: Delete Category.....	51
Figure 6.10: Edit Category.....	52
Figure 6.11: Products	Error! Bookmark not defined.
Figure 6.12: Add Product.....	52
Figure 6.13: Delete Product	Error! Bookmark not defined.
Figure 6.19: Main code	53

LIST OF TABLES

Table 5.2: Test Case i.....	42
Table 5.2: Test Case ii	42
Table 5.3: Test Case iii	43
Table 5.4: Test Case iv.....	43
Table 5.5: Test Case v.....	44
Table 5.6: Test Case vi.....	44
Table 5.7: Test Case vii	45
Table 5.8: Test Case viii	45
Table 5.9: Test Case ix.....	46
Table 5.10: Test Case x.....	46

CHAPTER NO 1 INTRODUCTION

1.1 Purpose

The purpose of this project, **BookNest**, is to develop a digital platform that streamlines the management and distribution of library books. This system is designed to replace traditional manual processes with a web-based solution that is efficient, secure, and easy to use for both administrators and users.

The system serves two main purposes:

1. **For Administrators:** It provides a dashboard where books can be added, updated, deleted, and PDF files can be uploaded with ease. This reduces manual record-keeping and ensures a centralized digital archive of books.
2. **For Users:** It offers a user-friendly website where anyone can browse books. However, in order to **download** any book, the user must first **register and log in**, ensuring controlled and secure access to digital content.

Overall, the system aims to:

- Improve accessibility to library books.
- Reduce the workload of library staff.
- Enable 24/7 access to digital books for students and readers.
- Promote a paperless, modern approach to library systems.

1.2 Project Scope

The scope of the **BookNest** project covers the complete development of a digital library system that enables administrators to manage books and users to access and download them through a secure and user-friendly web interface.

This system includes two major modules:

- **1. Admin Dashboard:**
 - Secure login for administrators.
 - Add, edit, and delete book records.
 - Upload and manage book PDFs.
 - View all uploaded books with essential details.
 - Manage the digital library without manual paperwork.

- **2. User Website:**
- Publicly accessible website interface.
- Display all available books with relevant information.
- Allow users to view book details.
- User registration and login functionality.
- Only authenticated users can download books.
- Redirection to login or registration if a non-logged-in user attempts to download.
- **Technologies and Functional Scope:**
- Web-based system accessible from any browser.
- Authentication and authorization for user access.
- PDF file handling for book downloads.
- Database integration for storing book and user data.

The project focuses on simplifying library operations and enhancing accessibility by offering a secure, scalable, and responsive platform for managing and accessing digital books.

1.3 Project Planning

The development of the **BookNest** project was carried out through a well-structured planning process to ensure smooth execution and timely completion. The project was divided into several key phases. Initially, the **analysis phase** involved gathering requirements for both the admin dashboard and the user website. This helped in identifying the core functionalities needed for managing books and user access.

Some milestones which we set for the project are the following:

Requirements Gathering

- **Design:** design for all types of users which is free from complexity.
- **Coding and Testing:** using the design and functional specifications the coding is done.
- **System testing:** this is the process of testing the web to verify it work according to specification.

1.3.1.1 Tools and Technologies

- **Frontend:** React.js for admin dashboard.

- **Backend:** Node.js, Prisma database.
- **Development Environment:** VS Code for code editing.
- **Version Control:** Git for source code management and collaboration.
- **Testing Tools:** Postman for API testing.

1.4 Technology We Used

1.4.1 React Native Technology

To develop the **BookNest** system, a range of modern web technologies was used to ensure that the application is responsive, efficient, and user-friendly. The frontend of both the user website and admin dashboard was built using **React.js**, a powerful JavaScript library known for its component-based architecture and fast rendering capabilities. Alongside React, **HTML** and **CSS** were used for structuring and styling the interface, while **JavaScript** added interactivity and dynamic behavior to the application.

On the backend, technologies such as **Node.js** and **Express.js** (*if used*) were considered for building RESTful APIs and managing server-side operations. The database used was **MongoDB** which stores all the essential information including user details, book records, and file paths for uploaded PDFs. For secure user authentication, **JWT (JSON Web Tokens)** or session-based methods were implemented, ensuring that only registered users could access and download books.

Version control was managed using **Git** and **GitHub**, and the project was developed in **Visual Studio Code**, which provided a robust and flexible development environment. These technologies together contributed to building a scalable, secure, and modern digital library system.

1.4.2 Hardware Requirement React JS

To develop a React Native app efficiently, the hardware must be capable of running development tools, simulators, and emulators smoothly. Below are the recommended hardware specifications:

- **Processor:** Intel Core i5 or higher (or an equivalent AMD processor) to ensure that development tools, compilers, and emulators run efficiently.
- **RAM:** At least 8 GB of RAM is necessary for running the development environment, but 16 GB or more is recommended for smoother multitasking and handling large projects.

- **Storage:** A minimum of 256 GB SSD is recommended to speed up file access, package installations, and build processes. Additional space is required to store app data, tools, and libraries.
- **Graphics:** Although a dedicated graphics card is not mandatory, it can improve performance, particularly when working with animations or resource-intensive emulators.
- **Operating System:**
 - **For Windows:** Windows 10 or later.
 - **For macOS:** macOS 10.14 (Mojave) or later (needed for iOS development).
 - **For Linux:** Any modern Linux distribution such as Ubuntu or Fedora.

This hardware setup ensures a smooth development experience for React Native, allowing you to build and test apps efficiently across both Android and iOS platforms.

1.4.3 Text Editor

First of all, you need a JavaScript-friendly text editor. Text editors are important when it comes to writing code. Depending on the features they provide, you can save hours of development time. There are some really good text editors out there with excellent language support. We are going to be using JavaScript in this book, so I'd recommend getting one of these open-source JavaScript-friendly text editors:

- Atom: <http://atom.io>
- Visual Studio Code: <http://code.visualstudio.com>

1.4.4 Node.js

Here's another important tool that we will be using throughout this book, Node.js. Node.js is a JavaScript runtime built on Chrome's V8 engine. It lets you run JavaScript outside your browser. Node.js has become popular because it lets you run JavaScript on the server and is really fast thanks to its non-blocking I/O methods. One other excellent advantage of Node.js is that it helps create command-line tools, which can be used for various purposes, such as automation, code scaffolding, and more, many of which we will be using in this book. At the time of writing this book, the latest Long Term Support (LTS) version of Node.js is 21.10.2. I'll be using this version throughout this book.

Once you have installed Node.js, run `node -v` in your Terminal (command prompt for Windows users) to check whether it is properly installed. This should print the current version of the node you have installed.

1.4.5 Google Chrome

Finally, in your system you should install the latest version of Google Chrome: <https://www.google.com/chrome/>. You can use Firefox or other browsers, but I will be using Chrome, so it will be easier for you to follow if you use Chrome.

Now that we have all the necessary tools installed in our system, let's get started with building our first application!

1.5 Book Nest

BookNest is a web-based library book management system designed to simplify and digitalize the process of managing and accessing library books. The system consists of two primary modules: an **Admin Dashboard** and a **Public Website**. The admin dashboard allows authorized administrators to manage the entire library system efficiently by adding, editing, or deleting book records, as well as uploading PDF files of books. On the user side, the website displays a collection of all available books. Users can view book details without logging in, but if they wish to download any book, they must first register and log in to the system. This login mechanism ensures that only authenticated users can access downloadable content, maintaining both security and control over the digital resources. BookNest aims to replace traditional, paper-based library systems with a modern digital solution that offers easy access, efficient management, and 24/7 availability. It is especially useful for educational institutions, students, and readers who prefer accessing books online in a secure and organized environment.

1.5.1 Admin Dashboard

The **Admin Dashboard** of **BookNest** serves as the control center for managing the entire digital library. It is accessible only to authorized administrators through a secure login system. The dashboard allows admins to perform essential operations such as **adding new books, editing book details, deleting records, and uploading PDF files** of books to the system. Each book entry contains important information like the title, author, category, and the attached PDF file for user access.

The interface is designed to be simple, clean, and easy to navigate, ensuring that even users with basic technical knowledge can manage the system effectively. The dashboard ensures that the digital library is always updated, accurate, and well-organized. By providing full control over the book database, the Admin Dashboard reduces manual effort, saves time, and enhances the efficiency of library management within the Book Nest system.

The application is designed to be user-friendly, secure, and efficient, aiming to simplify the process of booking and providing services while offering robust management tools for administrators.

1.6 Modules

1. Consumer Module.
2. Admin Dashboard

1.7 Benefits

The **BookNest** system provides several benefits to both administrators and users by transforming the traditional library model into a modern, digital solution. Its user-friendly interface and efficient functionality make library operations faster, simpler, and more accessible.

- **Efficient Book Management:** Admins can easily add, edit, delete, and upload books through a centralized dashboard.
- **Time-Saving:** Reduces the need for manual record keeping and paper-based systems.
- **Real-Time Updates:** Any changes made by the admin are instantly reflected on the website.
- **Secure Access Control:** Only authorized users can access admin functionalities.
- **For Users:**
 - **Easy Book Access:** Users can browse and view all available books anytime from any device.
 - **Secure Downloads:** Only registered and logged-in users can download books, ensuring controlled access.
 - **Improved User Experience:** Clean, responsive interface makes browsing and searching for books simple.
 - **24/7 Availability:** Users can access the library at their convenience without depending on physical timings.
- **Overall System Benefits:**
 - **Digital Transformation:** Moves the library system from manual to digital, promoting eco-friendliness.
 - **Scalability:** Can easily be expanded to include more features or support a larger user base.
 - **Accessibility:** Makes books available to users regardless of their location

1.7.1 Benefits for Consumers

The **BookNest** system offers several valuable benefits to consumers, particularly students, readers, and library users who prefer accessing digital content. One of the main advantages is **easy and instant access to books** through a user-friendly website. Consumers can browse the entire collection without any restrictions and download their desired books after a simple registration and login process. This ensures that digital resources are accessible anytime, anywhere, without the need to visit a physical library.

The system also ensures **secure and personalized access**, allowing users to manage their own accounts and keep track of downloaded materials. With a clean and responsive interface, the platform provides a smooth user experience across different devices. Additionally, the availability of downloadable PDFs promotes convenience and flexibility for users who want to study offline.

1.7.2 Benefits for Administrators

The **BookNest** system provides significant benefits for administrators by simplifying and automating the process of managing library resources. Through the secure Admin Dashboard, administrators can easily perform core tasks such as adding new books, editing existing records, deleting outdated entries, and uploading PDF files — all from a single, centralized platform. This not only saves time but also minimizes the chances of manual errors.

The system ensures **real-time updates**, so any changes made by the admin are instantly reflected on the user-facing website. It also offers a secure environment, allowing only authorized personnel to access the dashboard, thereby preventing unauthorized modifications. With an intuitive interface, even non-technical staff can operate the system efficiently. Overall, BookNest empowers administrators to maintain an organized, accurate, and up-to-date digital library with minimal effort and maximum control.

1.8 Objective

The objective of the **BookNest** project is to design and develop a web-based library management system that facilitates both administrators and users in handling digital books effectively. The system aims to allow administrators to manage book records by adding, editing, deleting, and uploading book PDFs through a secure dashboard. On the user side, the objective is to provide a simple and accessible interface where users can browse available books and download them after logging in. The project also focuses on implementing a secure authentication system to ensure that only registered users can access downloadable content. Overall, the main goal is to streamline library operations, enhance accessibility, and offer a digital solution that is efficient, secure, and user-friendly.

1.9 Feature

1.9.1 Admin Feature

- Secure Login: Only authorized admins can access the dashboard.
- Add Books: Admins can add new books with title, author, category, and PDF file.

- **Edit Book Details:** Update existing book information and replace uploaded files.
- **Delete Books:** Remove unwanted or outdated books from the system.
- **Upload PDFs:** Attach and manage downloadable PDF files for each book.
- **Manage Book Listings:** View all uploaded books in an organized layout

1.9.2 Website Feature

- Users can **browse a wide range of books** organized by category, title, or author.
- Users can **view book details**, such as title, author, description, and category.
- Books are available in **PDF format for download or reading online**.
- The website offers a **clean and user-friendly interface** for easy navigation.
- Responsive **design** ensures accessibility on desktops, tablets, and mobile devices.
- Search **functionality** allows users to quickly find books by keywords.
- A **contact or support section** may be available for queries or feedback.
- Visitors can **access the platform without registration**, enabling hassle-free use.

CHAPTER NO 2 BACKGROUND

2.1 Background Research

In recent years, the demand for digital learning and online reading platforms has increased significantly due to the rapid advancement of technology and internet accessibility. Traditional bookstores are gradually being replaced by online platforms that offer convenience, variety, and instant access to books. Many students and readers prefer digital libraries for their affordability, availability, and ease of use. Recognizing this shift, we researched various existing online book platforms to understand their features, limitations, and user expectations. This research helped us identify the need for a simple, user-friendly website where users can access and read books without the need for a complex registration process. Our project, **Book Nest**, aims to bridge this gap by offering an efficient and accessible platform for book lovers and learners.

2.2 Existing Technology

Several technologies currently support the development and operation of online bookstores and digital reading platforms. Frontend technologies like **HTML**, **CSS**, and **JavaScript** are widely used to design responsive and interactive user interfaces. **React.js** and **Vue.js** offer powerful frameworks for building modern, component-based user experiences. On the backend, technologies such as **Node.js**, **Express.js**, and **PHP** enable efficient server-side logic and API development. For data storage and management, **MongoDB**, and **Firebase** are commonly used. Additionally, platforms like **Open Library**, and **Google Books** utilize advanced features including cloud hosting, recommendation algorithms, and user analytics to enhance user engagement. These technologies provide a strong foundation for creating an efficient and scalable digital bookstore like **Book Nest**, combining performance, accessibility, and ease of maintenance.

2.3 Reason of Project

- Make things simple
- Easy to use
- Security
- User Friendly
- Time saving

2.4 Tools

Tools for the project are as follows:

2.4.1 React

React is a JavaScript library created for building fast and interactive user interfaces for web and mobile applications. It is an open-source, component-based, front-end library responsible only for the application's view layer. In Model View Controller (MVC) architecture, the view layer is responsible for how the app looks and feels. React was created by Jordan Walke, a software engineer at Facebook.

2.4.1.1 Why React?

React's popularity today has eclipsed that of all other front-end development frameworks. Here is why:

- Easy creation of dynamic applications: React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly.
- Improved performance: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM compares the components' previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.
- Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their logic and controls, and they can be reused throughout the application, which in turn dramatically reduces the application's development time.

2.4.1.2 Performance

React uses VDOM, which makes the web applications run much faster than those developed with alternate front-end frameworks. React breaks a complex user interface into individual components, allowing multiple users to work on each component simultaneously, thereby speeding up the development time.

2.4.2 Node.js

Node.js is a technology to build the back-end of applications with JavaScript. Google Chrome's V8 JavaScript engine powers Node.js. Node.js is effective and lightweight because it uses an event-driven and non-blocking I/O model. Developers use this framework for hosting APIs, serving HTTP requests, and accessing the database.

2.4.2.1 Who uses Node.js?

Currently, Node.js is used by many companies and has several famous clients, like Netflix, Trello, and Uber. Below we share why Node.js has become their framework of choice.

Netflix, the leader of online video streaming, conducts A/B testing to provide 93 million users with a rich experience. The platform faces problems of conditional dependencies and app scalability. To solve these issues, the company has chosen Node.js for its fast speed and that it's lightweight. In this way, Netflix has managed to decrease startup time by 70%.

Trello, a project management app, uses Node.js for its server-side. The Trello team found Node handy for system updates, which need a lot of open connections.

2.4.2.2 Node JS Key Features

Now, let us find out about the main Node.js features.

- Open-source. Node.js is a free and open-source framework.
- Enhanced performance. Developers can perform non-blocking operations, which enhances web app performance.
- Server development. Node has in-built APIs. Consequently, Node.js allows developers to make different servers like DNS servers, TCP server, HTTP server, and more.
- Unit testing. Node.js has unit testing called Jasmine, which allows testing ready code quickly.
- Scalability. Apps built with Node.js can be scaled in both a Vertical and Horizontal way to improve their performance.

2.4.3 MongoDB

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

2.4.3.1 MongoDB Features

1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

2.4.3.2 Why Use MongoDB?

1. Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
2. Ad hoc queries - MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
3. Indexing - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.

4. Replication - MongoDB can provide high availability with replica sets. A replica set consists of two or more Mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.
5. Load balancing - MongoDB uses the concept of sharing to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

2.5 Tool Used to Develop “Website”

2.5.1 Visual Studio Code

Visual Studio Code is Microsoft’s tool for source code editing and it is designed for Windows, Linux. It’s a tool used as a support for embedded git control, debugging, syntax highlighting snippets, intelligent code completion, and code refactoring. It is an agile, cross-platform, and multi-language editor, which you can download directly from the Visual Studio website. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

2.5.1.1 Why VS Code?

What makes VS Code special? Microsoft’s Visual Studio Code is a text editor with powerful IDE-like features. Side note: VS Code is a separate product from Visual Studio the IDE. In terms of what is available right now, it seems like VS Code is a “best of both worlds” option between IDEs and Text Editors. Here are some more reasons why we’re using VS Code:

- VS Code is simple to download and install on any OS.
- It is completely free.
- It is non-opinionated and open source, with non-proprietary standards.
- You can bring your favorite key bindings from Sublime, Vim, etc. with you.

- It's easy to customize and has many useful extensions.
- It has powerful IDE-like features including a built-in debugger and terminal.

2.5.1.2 Debugger

The built-in VS Code Debugger for Node.js is very powerful, and it can be expanded with the popular Chrome Debugger extension. However, this requires a server to be running and is outside the scope of this blog post (perhaps in the future we can have a blog post about Debugging in VS Code).

That said, if you are still interested, check out articles for debugging Node [here](#) and for React [here](#). And here is the full reference for Node debugging.

CHAPTER NO 3 SYSTEM REQUIREMENTS ANALYSIS

3.1 Introduction of the Requirement

In this chapter, I briefly explain what a software requirements specification is how it is gathered from the client and what challenges we face in gathering requirements. In this chapter we talk about the requirements of Book Nest Functional and Non-functional.

3.2 Software Requirement Specification (SRSs)

A software requirements specification (SRS) is a description of the software system to be developed. It specifies functional and non-functional requirements and may include a set of use cases that describe the user interactions that the software must provide.

The software requirements specification forms the basis for agreement between customers and suppliers or vendors (marketing and development divisions may play these roles in market-driven projects) about what the software product is expected to do and not do. A software requirements specification allows rigorous assessment of requirements before design begins and reduces rework later. It should also provide a realistic basis for estimating product costs, risks and schedules. When used correctly, software requirements specifications can help prevent software project failure.

A software requirements specification document contains enough of the necessary requirements that are required to develop a project. To derive requirements, the developer must have a clear and thorough understanding of the products to be developed or developed. This is achieved and refined through detailed and continuous communication with the project team and customer until the software is completed.

3.3 Gathering requirements from client

The Software Requirements Specification (SRS) is a communication tool between stakeholders and software designers. The specific goals of the SRS are:

- Facilitating reviews
- Describing the scope of work
- Providing a reference to software designers (i.e. navigation aids, document structure)
- Providing a framework for testing primary and secondary use cases

- Including features to customer requirements
- Providing a platform for ongoing refinement (via incomplete specs or questions)
- Reliability Availability Security Maintainability Portability.

3.4 Typical Requirements Gathering Process

Our consultant is going to talk to a client about a new Intranet. The client already has an Intranet, but wants a new one. It works well in the main, but the technology is old and the client would like some new features. Over the course of several days our consultant quizzes the client about all areas of the system. He is a good consultant, so he does the following:

- Structures the meeting into particular topic areas (review of old system, interview with users, potential new features etc.) and works through them with the appropriate people.
- Takes notes from all his meetings.
- Spends some time himself working through the system, forming his own impressions.
- Writes up a report at the end of the requirements phase, with his recommendations. Our consultant reports back to his project team that he understands the problem, works with a developer to devise a solution, and sits back with a rosy feeling as the solution is built.
- Client thinks about a requirement
- Client communicates a requirement
- Consultant hears requirement
- Consultant documents requirement

3.5 Capture Better Requirements

So how do we mitigate against these factors, if they are oh so inevitable in requirements gathering? Good question. Try the following tips. They won't magically solve the problem, but they will all help you keep the percentage of the original requirement you record as high as possible. There are eight points to gather better requirement:

1. Firstly, accept you will never hit 100% accuracy. Bear it in mind at all times and it will drive you to capture better requirements.

2. Spend as long as you possibly can on this phase and talk to as many people as possible. Talk to people in a wide range of roles, but ask them all the same questions. You will be surprised how good an insight staff “on the shop floor” have about so called “management level” questions (and vice versa).
3. Ask “Why?” Ask it a lot. Never accept an answer in the requirements gathering phase without asking “why” at least five times. No matter the project, the technology, the company or the interviewee -- “Why?” is the best question.
4. Requirements gathering has to be conducted by as few people as possible. Imagine a different consultant carrying out each meeting in the scenario described above. Our percentage would plummet. If more than one person is really needed (and avoid it if you can), then the process has to be headed up and coordinated by a single person. Some single person must hold the whole problem in their head.
5. Always have a dedicated scribe. In meetings and workshops, the person conducting the session should never write the notes. If they are doing a good job (see the first point), then they just won’t have time.
6. Better still, use “dialogue mapping” techniques to document meetings. There is not enough room to go into this topic here, but this involves a dedicated scribe documenting using diagrams on a big screen that everyone can see.
7. Deliver outputs of requirements gathering to the client as often as possible. Always send them a copy of your raw meeting notes. Then follow up meetings with your findings. If you need to produce a formal report, go over your thinking and draft copies with the client as early as possible.
8. Your final report will not be perfect, so when development starts, keep talking to the client. If possible, keep showing them the solution as you build it to check your theories and ideas.

3.6 Different Types of Software Requirements

3.6.1 Business Requirements (BR)

- These are high-level business goals of the organization building the product, or the customer who commissioned the project.
- These are usually provided as a single page of high-level bullets.

3.6.2 Market Requirements (MR)

- These drill down into BRs, but still are high-level. In addition to business goals, they also outline market needs.
- These are usually provided as a prioritized bulleted list or table, and are usually less than 5 pages long.

3.6.3 Functional Requirements (FR) – Use Cases

- These cover the functionality of the product in detail. Use cases are one of the best ways of documenting functional requirements.
- Depending on the product being built, FRs can run several hundred pages.

3.6.4 Non-Functional Requirements (NFR)

- These are not related to the “functionality” of the product – but cover goals such as Reliability, Scalability, Security, Integration, etc.
- Many projects make the mistake of not specifying these explicitly.

3.6.5 UI Requirements (UIR)

- User interface specs are not considered “requirements” in traditional requirements management theory.
- Phooey! In my opinion, UI specifications are really requirements (what else are they?) - And in fact, they should be considered an integral part of the requirements for any software that has a user interface.

3.7 Requirements for Book Nest

The success of the Book Nest online bookstore depends on fulfilling both **functional** and **non-functional** requirements. These requirements ensure the system operates efficiently, provides a smooth user experience, and meets the goals of administrators, customers, and vendors.

3.7.1 Functional Requirements

In Computer Science and systems engineering, a functional requirement defines the function of a system or its components. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

3.8 Functional Requirement of BookNest Website:

- Order Booking
- Select an item from the menu
- Add an item to their current order
- Review their current order
- Remove an item/remove all items from their current order
- Admin Confirmation
- Add/Remove/Edit Products from inventory
- Add categories

3.8.1 Order Booking

- **User Registration and Login:**

Users can register and log in using their credentials to access personalized features like order history and profile management.

- **Book Browsing and Search:**

Customers can browse books by categories (e.g., Fiction, Non-Fiction, Academic), or use the search bar to find specific titles, authors, or genres.

- **Book Detail View:**

Users can click on a book to view its details, including cover image, description, author, price, and availability.

- **WhatsApp Inquiry Integration:**

Instead of a shopping cart, users can contact the seller directly via WhatsApp for more information or to place an order.

- **Admin Dashboard Login:**

Admins have a secure login to access the dashboard for managing the website.

- **Book Listing Management:**

Admins can add, edit, or delete book listings with fields such as title, description, price, category, and image.

- **Category Management:**

Admins can create and manage different book categories for easier navigation.

- **Customer Inquiry View:**

Admins can view incoming WhatsApp inquiries or messages related to books.

- **Responsive Design:**

The website is responsive and works efficiently on desktop, tablet, and mobile devices.

- **Search Engine Friendly:**

URLs, content, and layout are optimized for SEO to increase visibility.

3.8.2 Add/Remove/Edit Products from inventory

Admin can add, remove or edit products from the app's inventory.

3.8.3 Add categories

Admin can make categories for keeping the products well-organized.

3.9 Non-Functional Requirement of Book NEST

- **Performance:**

The website should load quickly and respond to user actions within 2–3 seconds.

- **Scalability:**

It should be able to handle more users and book listings in the future without performance issues.

- **Availability:**

The system must be available 24/7 with minimal downtime.

- **Usability:**

The interface should be easy to use for all types of users.

- **Security:**

User and admin data must be protected with secure login and HTTPS protocol.

- **Compatibility:**

The website should work well on all major browsers and devices.

3.9.1 Easy to understand Interface for any non-technical user

Easy is something that can be done with ease, is not hurried, is not difficult, is free from anxiety or is comfortable. An example of easy is cooking a meal without the pressure of time. An example of easy is a hike that continues at the same elevation.

A user interface specification (UI specification) is a document that captures the details of the software user interface into a written document. The specification covers all possible actions that an end user may perform and all visual, auditory and other interaction elements.

But if no one can figure out to how to move beyond the home screen, it's not going to sell well. No matter how flashy an app looks, what most people want is an app that does exactly.

3.9.2 Reliability

Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs according to its specifications. It has long been considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability - RAS, for short - are considered to be important aspects to design into any system. In theory, a reliable product is totally free of technical errors; in practice, however, vendors frequently express a product's reliability quotient as a percentage. Evolutionary products (those that have evolved through numerous versions over a significant period of time) are usually considered to become increasingly reliable, since it is assumed that bugs have been eliminated in earlier releases. For example, IBM's z/OS (an operating system for their S/390 server series), has a reputation for reliability because it evolved from a long line of earlier MVS and OS/390 operating system versions.

3.9.3 Security

Have you ever heard the old saying “You get what you get and you don’t get upset”?

While that may apply to after school snacks and birthday presents, it shouldn't be the case for software security. When a software feature is deployed, it isn't simply accepted by the software owner; there's a strategic process of critique, justification, and analysis before its deployed.

Security should be treated with the same attention to detail. After all, secure software doesn't just happen out of nowhere it has to be a requirement of the strategic development process. The requirements should be clear, consistent, testable, and measurable to effectively deploy secure software.

Traditionally, requirements are about defining what something can do or be. A hammer has to be capable of driving nails. A door lock needs to keep a door closed until it's unlocked with a specific key. A car needs to move travelers from point A to point B along the nation's roads. It also needs to work with the modern gasoline formulation. These types of requirements work fine for physical objects, but fall short when designing software.

3.9.4 Usability

Usability is the measure of a product's potential to accomplish the goals of the user. In information technology, the term is often used in relation to software applications and Web sites, but it can be used in relation to any product that is employed to accomplish a task (for example, a toaster, a car dashboard, or an alarm clock). Some factors used in determining product usability are ease of use, visual consistency, and a clear, defined process for evolution.

CHAPTER NO 4 SYSTEM DESIGN

4.1 Methodology

The development of the **BookNest** website followed the **Waterfall Model** of software development. This model is a linear and sequential approach, where each phase is completed before the next one begins. Initially, a detailed requirement analysis was conducted to gather both functional and non-functional requirements from users and stakeholders. Based on these requirements, the system design was created, which included the layout of the website, database schema, and overall flow of the application. After the design phase, the actual implementation began using modern technologies such as React.js for the frontend and Node.js with MongoDB for the backend. Once development was completed, the website underwent thorough testing to identify and fix any bugs or issues. Finally, the website was deployed and made live for users. Post-deployment, regular maintenance and updates are performed to ensure smooth functioning and to incorporate user feedback.

4.2 Unified Modeling Language

The Unified Modeling Language (UML) was created to forge a common, semantically and syntactically rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally. UML has applications beyond software development, such as process flow in manufacturing. It is analogous to the blueprints used in other fields and consists of different types of diagrams. In the aggregate, UML diagrams describe the boundary, structure, and behavior of the system and the objects within it. UML is not a programming language but some tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. Actors represent roles that users take on when they use the system and the role of a check-in employee. Use cases are to describe interactions, and flows of batch processing, which generally does not include interactions, and can also be described as use cases.

4.3 Designing Technique

A designing technique supported by the software design people defines how a particular task is to be performed. Design technique is to support design work, the aims of which can be varied, though they may include gaining key insights or unique essential truths resulting in more holistic solutions.

4.4 Waterfall Model

The waterfall model is a sequential (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/ implementation and maintenance. Despite the development of new software development process models, the Waterfall method is still the dominant process model with over a third of software developers still using it. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases, do not overlap.

4.4.1 Why we used waterfall model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.
- Requirements are completed early in the project, enabling the team to define the entire project scope, create a complete schedule, and design the overall application.
- In this model phases are processed and completed one at a time. Phases do not overlap.

4.4.2 Diagram of Waterfall Model

The software development process illustrates in the Waterfall Model which is shown in Fig:

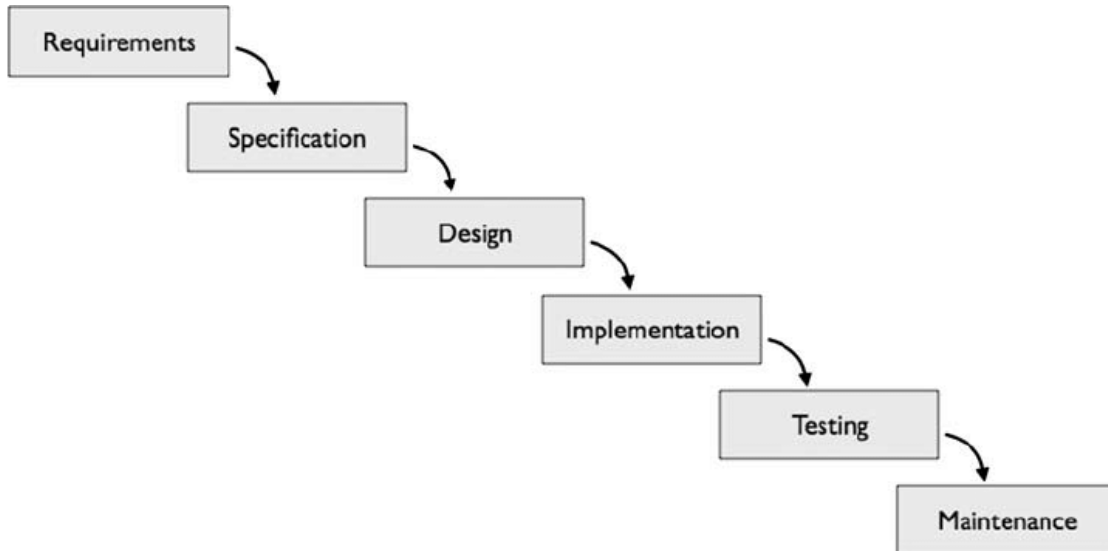


Figure 4.1: Waterfall Model

We use the Waterfall model to develop the Book Nest React web it is much more reliable to develop an Android project. It is an old development model it's include all the feature to develop a large scale project or product.

4.5 System Features

4.5.1 System Features 1

- **User Registration and Login**
Allows users to create accounts and securely log in to access personalized features.
- **Admin Dashboard**
Enables the admin to manage book listings, users, orders, and content through a centralized panel.
- **Book Browsing and Search**
Users can browse books by categories, authors, or use the search bar to find specific titles.
- **WhatsApp Contact Integration**
Instead of a cart, users can directly contact the seller via WhatsApp for inquiries or orders.
- **Mobile Listing Management**
Admin can easily add, edit, or delete mobile/book listings from the backend.

4.5.2 System Features 2

- The Website will be user friendly.
- The application should be able to complete its operation within reasonable response time.
- This website will support React advanced version with latest database with the help of backward compatibility.

4.6 Use case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involve.

UCDs have only 4 major elements: The actors that the system you are describing interacts with, the system itself, the use cases, or services, that the system knows how to perform, and the lines that represent relationships between these elements.

There are two different types of use cases: business use cases and system use cases. A business use case is a more abstract description that's written in a technology-agnostic way, referring only to the business process being described and the actors that are involved in the activity.

The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. Create a professional diagram for nearly any use case using our UML diagram tool.

The actors that the system you are describing interacts with, the system itself, the use cases, or services, that the system knows how to perform, and the lines that represent relationships between these elements. Create a professional diagram for nearly any use case using our UML diagram tool.

4.6.1 Use case 1:

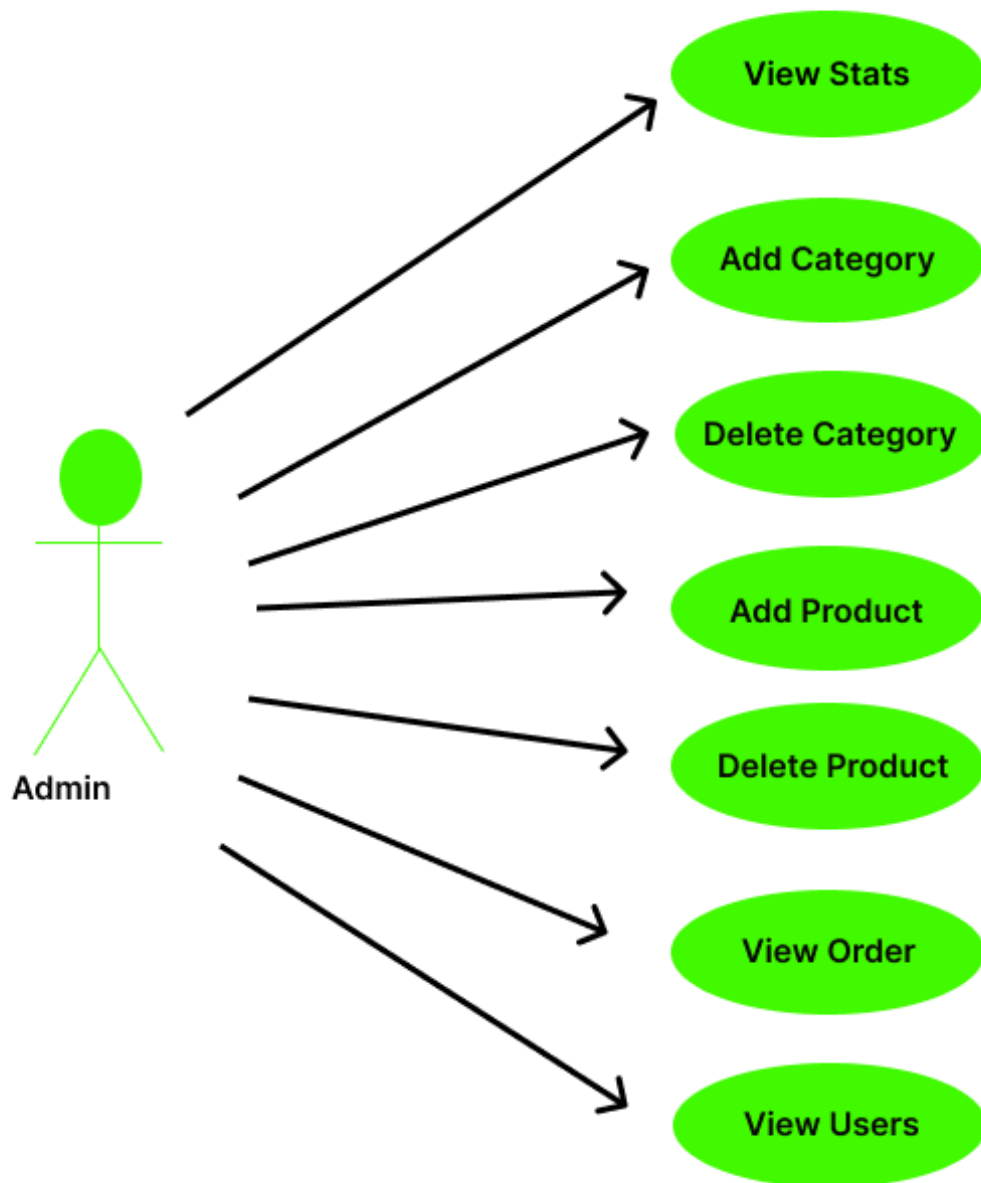


Figure 4.2: Use case of Admin

4.6.2 Use case 2:

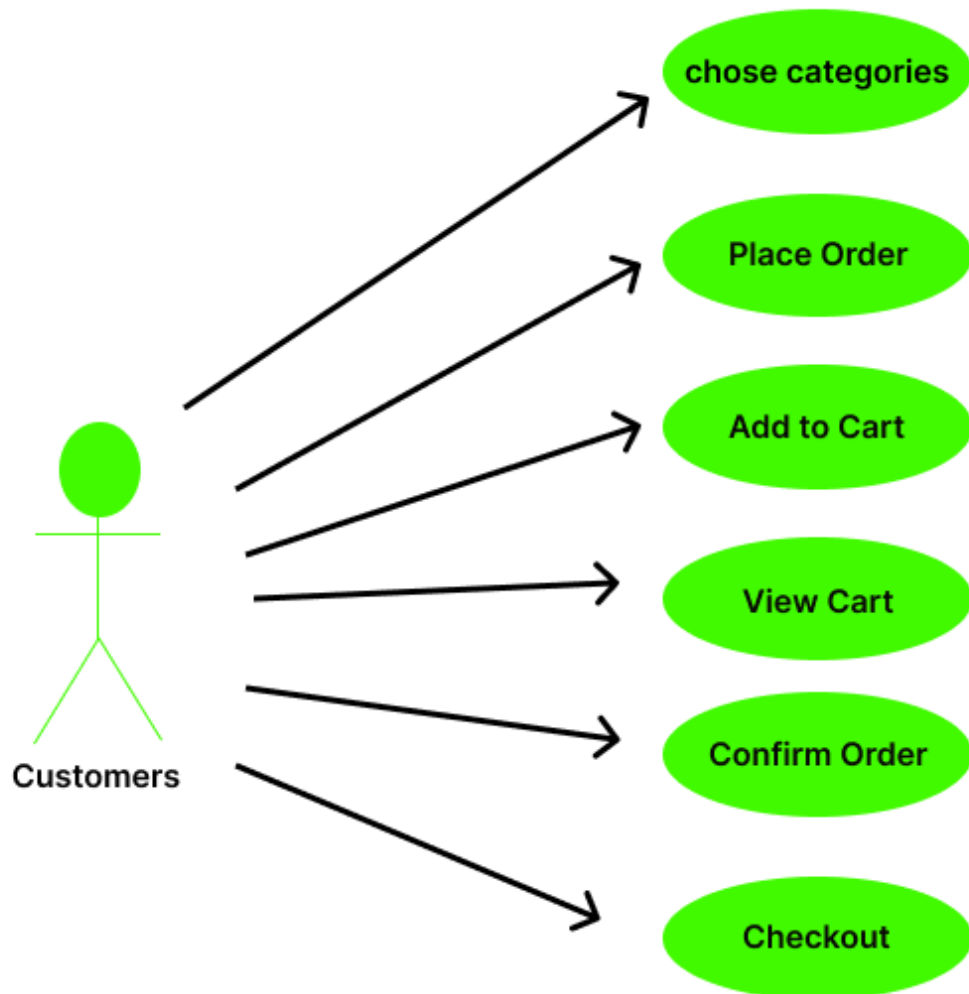


Figure 4.3: Use case of Admin

4.7 Entity Relation Diagram

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

4.7.1 The elements of an ERD are:

- Entities
- Relationships
- Attributes

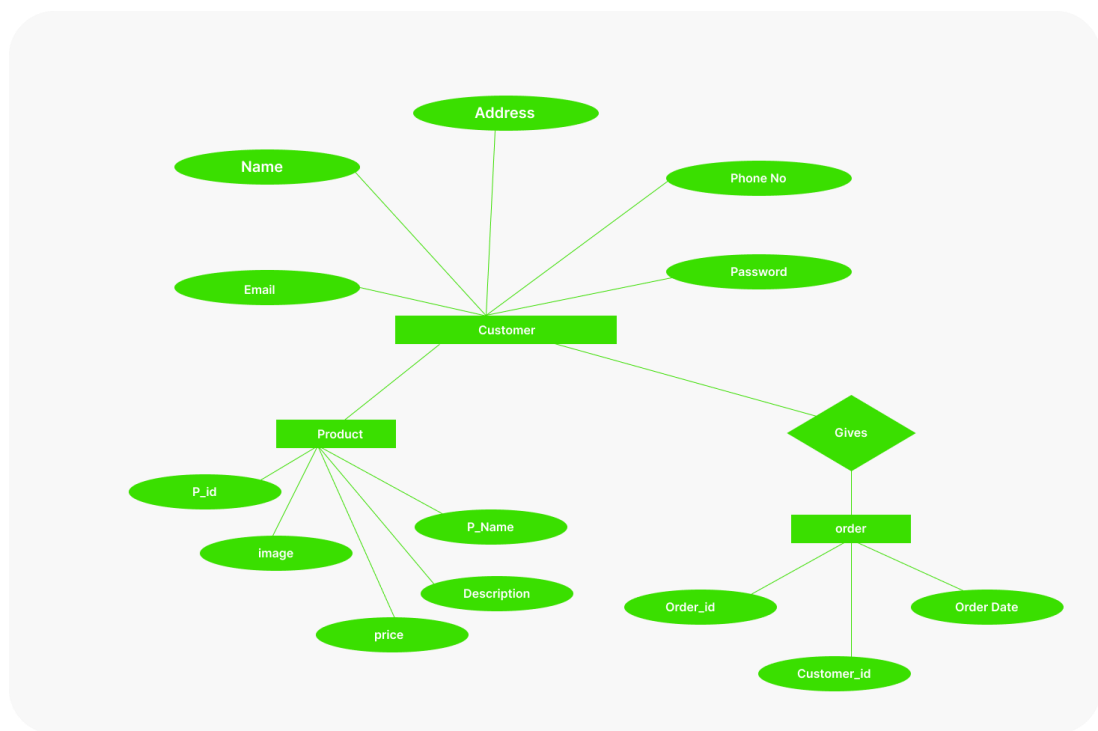


Figure 4.4: Entity Relation Diagram

4.7.2 Steps involved in creating an ERD include:

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analyzing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD

4.8 System Sequence Diagram

In Computer Science, a system sequence diagram (SSD) is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events.

4.8.1 Overview

System sequence diagrams are visual summaries of the individual use cases. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

- A system sequence diagram should specify and show the following:
- External actors
- Messages (methods) invoked by these actors
- Return values (if any) associated with previous messages
- Indication of any loops or iteration area

4.8.2 Advantages of sequence diagram

- UML sequence diagram helps you to envision what will happen during the execution of a use case
- They are great to help developers and business analysts get to a common understanding.

4.8.3 Sequence Diagram

It shows all scenarios and possible situation which performed using Book Nest website. Due to this diagram any new person can easily understand how user interact the app

and how Book Nest response to the users.

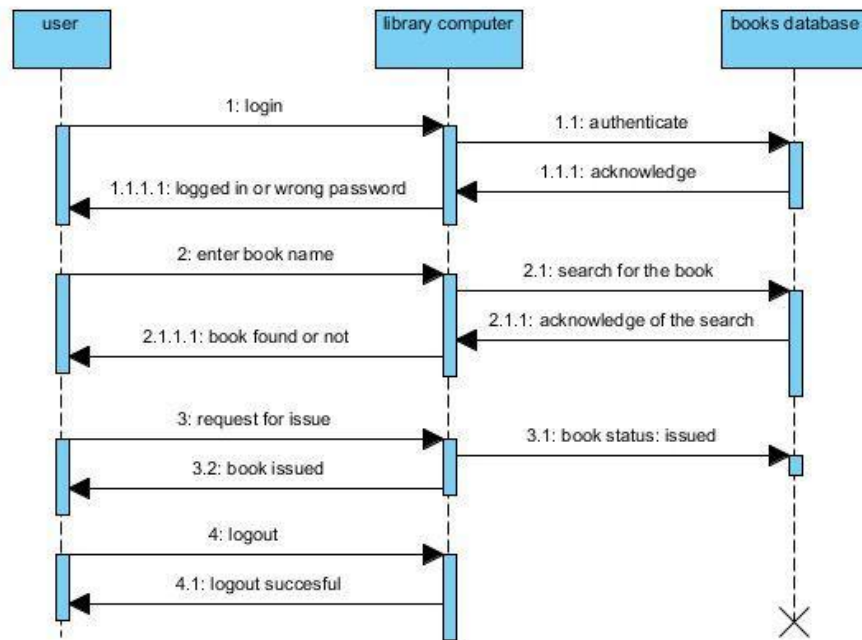


Figure 4.5: System Sequence Diagram

CHAPTER NO 5 TESTING & IMPLEMENTATION

5.1 Project Testing

Testing is the process of executing the program to find out errors. During testing, the program to be tested is executed with a set of test cases, and the output of the programs for the test case is evaluated to determine if the program is performing as it is expected to. The success of testing in revealing program errors depends critically on the test cases. Also, the use of testing is not limited to the testing phase. The results of testing are used later on during maintenance. During testing, a test suite can be used to see that the modification doesn't have any undesirable effect.

5.2 Purposes of testing

Software testing enables making objective assessments regarding the degree of conformance of the system to stated requirements and specifications. Testing verifies that the system meets the different requirements, functional, performance, reliability, security, usability, and so on. This verification is done to ensure that we are building the system right. In addition, testing validates that the system being developed is what the user needs. In essence, validation is performed to ensure that we are building the right system. Apart from helping make decisions, the information from software testing helps with risk management. Software testing contributes to improving the quality of the product. You will notice that we have not mentioned anything about defects/bugs up until now. While finding defects/bugs is one of the purposes of software testing, it is not the sole purpose. It is important for software testing to verify and validate that the product meets the stated requirements/specifications. Quality improvements help the organization to reduce post-release costs of support and service while generating customer goodwill that could translate into greater revenue opportunities. Also, in situations where products need to ensure compliance with regulatory requirements, software testing can safeguard the organization from legal liabilities by verifying compliance.

5.3 Benefits of Automated Software Testing

Test Automation is essentially using code to create programs that perform automated tests for your software. The way this is different from manual testing is, that instead of

performing the test, one creates an automated testing scenario and supervises it. Test automation is extensively used for regression testing, which seeks out new bugs in a program and separates them. Regression tests are generally extremely tedious and time-consuming. Here is where automated tests come in and make life easy for a software testing professional. Apart from this code-driven testing type, the other arena for automated testing is user environment simulation. Testing software can be created to replicate a typical user environment using automated keystrokes and mouse clicks. The software GUI response is recorded and analyzed as per the automated input.

5.3.1 Efficient Testing

Test automation is a way to make the testing process extremely efficient. The testing team can be strategically deployed to tackle the tricky, case-specific tests while the automation software can handle the repetitive, time-consuming tests that every software has to go through. This is a great way to not only save time, money, and resources but also generate a high ROI.

5.3.2 Upgradation and Reusability

One of the best aspects of test automation is that the testing software is reusable. Not only that but with every new test and every new bug discovery, the testing software directory can be upgraded and kept up-to-date. Thus, even though one of the main criticisms against test automation is the expense, one has to realize that automation software is a long-lasting, reusable product that can justify its cost.

5.3.3 Consistency

Test automation provides a consistent platform for your testing needs. The tests for which automation is usually deployed are extremely tedious. Automation drastically reduces the margin of error in the testing scenario by going through pre-recorded instructions.

Regression tests verify whether the pre-existing functionalities are suited for new versions, which is critical when new development in the existing software takes place. This novel consistency provides much-needed reliability for your testing protocols. Thus, even though one of the main criticisms against test automation is the expense, one has to realize that automation software is long-lasting.

5.3.4 Unique Programmability and Shelf Life

Not only can test automation software be built to exact testing specifications, but it also serves as a prime component for future testing scenarios. In-house automated software developed by testing firms is modeled such that they have enough flexibility to handle a unique product while complying with the latest security and testing protocols. This makes test automation a powerful tool for time-saving, resourceful, and top-notch results.

5.3.5 User Environment Simulation

One unique way in which testing automation affects the testing procedure is through the simulation of a typical user environment through categorically deployed mouse clicks and keystrokes. GUI testing is one of the most time-consuming and redundant procedures because the tester has to deploy the same procedures in mock user-driven environments and check for issues in the responsiveness of the GUI. With automated testing, this process becomes incredibly easy.

Get in touch with Optimus QA for more information regarding automated testing. As we saw above, automated testing can make things in the testing lab very easy and tester-friendly. At the same time, it gives the software creates the unique advantage of quick, hassle-free testing and a great way to save precious resources and stay ahead of the timeline.

5.4 The Basic Levels of Testing are

- User Needs Acceptance Testing
- Requirements System Testing
- Design Integration Testing
- Code Unit Testing
- Levels of Testing
- System verification testing
- Alpha Testing
- Beta Testing

5.5 Users need acceptance Testing

User acceptance is a type of testing performed by the Client to certify the system concerning the requirements that were agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment. The main purpose of this testing is to validate the end-to-end business flow. It does NOT focus on cosmetic errors, Spelling mistakes, or System testing. This testing is carried out in a separate testing environment with a production-like data setup. It is a kind of black box testing where two or more end users will be involved.

5.5.1 Who performs user acceptance testing?

- Client
- End Users

5.6 Prerequisites of User Acceptance Testing

The following are the entry criteria for User Acceptance Testing:

- Business Requirements must be available.
- Application Code should be fully developed.
- Unit Testing, Integration Testing & System Testing should be completed.
- No Showstoppers, High or medium defects in the System Integration Test Phase.
- Regression Testing should be completed with no major defects.
- All the reported defects should be fixed and tested before UAT.
- The traceability matrix for all testing should be completed.
- Sign off mail or communication from the System Testing Team that the system is ready for UAT execution.

5.7 System Testing

The process of testing an integrated system to verify that it meets specified requirements.

5.8 Analogy

During the process of manufacturing a ballpoint pen, the cap, the body, the tail, the ink

cartridge, and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed.

5.8.1.1 Method

Usually, the Black Box Testing method is used.

5.8.2 System Test Plan

- Prepare
- Review
- Rework
- Baseline
- System Test Cases
- Prepare
- Review
- Rework
- Baseline
- System Test
- Perform

5.8.3 When is it performed?

System Testing is performed after Integration Testing and before Acceptance Testing.

5.8.4 Who performs it?

Normally, independent Testers perform System Testing. Performs testing.

5.9 Unit Testing

The level of testing is called unit testing. In this, different modules are tested against the specifications produced during the design of the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goal is set to test the internal logic of the modules.

A unit is the smallest testable part of the software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual

program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class, or derived/ child class. Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module. Unit testing frameworks, drivers, stubs, and mock/fake objects are used to assist in unit testing.

5.9.1 Who performs it?

Unit Testing is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.

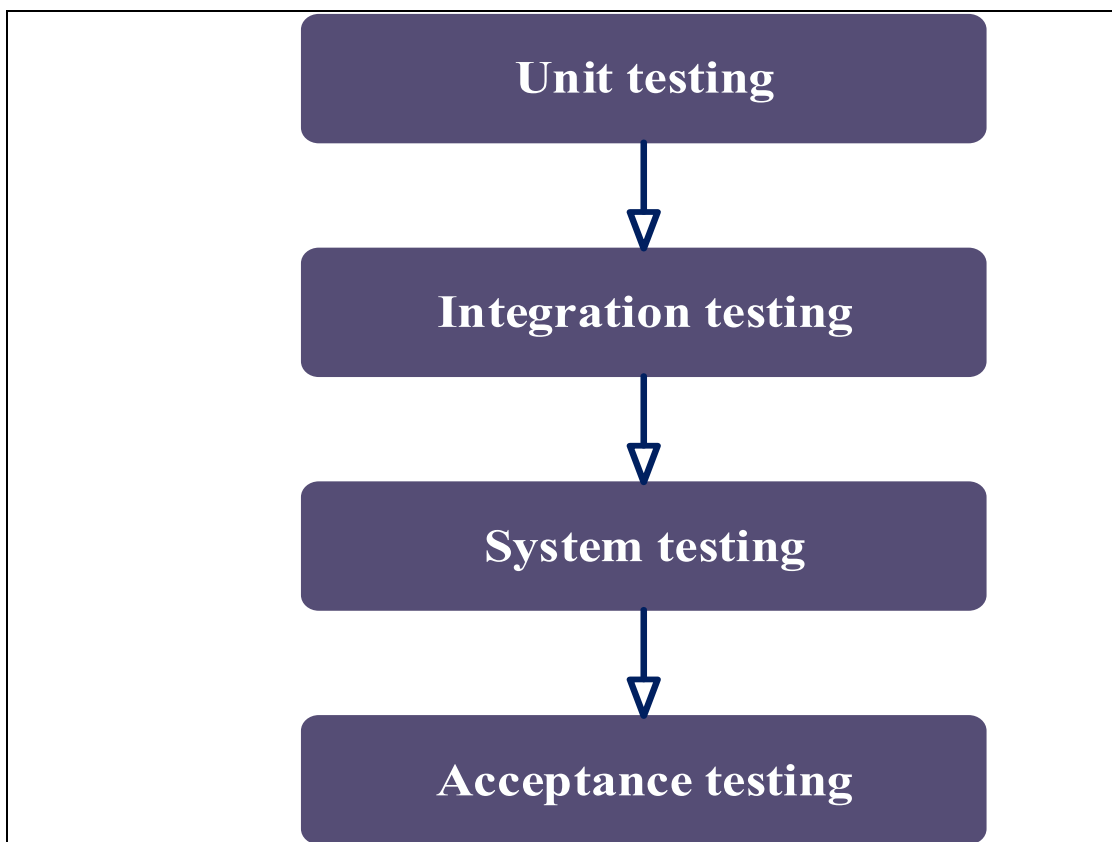


Figure 5.1: Testing Diagram

5.10 Integration Testing

The next level of testing is often called integration testing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if the modules can be integrated properly, the emphasis being on the testing interface between modules. This testing activity can be considered a testing design, hence the emphasis on testing module interactions.

Integration testing: Testing performed to expose defects in the interfaces and the interactions between integrated components or systems.

Integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components.

Integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

5.11 System Testing

During system testing, the system is used experimentally to ensure that the software doesn't fail, i.e. it will run according to its specifications and in the way, users expect, special test data input for processing, and the results examined. A limited number of users may be allowed to use the system and can see whether they try to use it in unforeseen ways.

5.12 Acceptance Testing

It is sometimes performed with realistic data of the user to demonstrate that the software is working properly. Testing here focuses on the external behavior of the system. The internal logic of the program is not emphasized.

5.13 Validation Check

During the testing section, validation checks are made. Appropriate actions are taken after testing.

5.14 Implementation

Implementation is the stage of the project when the theoretical design is turned into a working system. At this stage, the main workload, the upheaval, and the major impact on the existing practices shift to the user department. If the implementation stage is not carefully planned and controlled, it can cause chaos. Thus it can be considered to be the most crucial stage in achieving a new successful system and in giving the users confidence that the new system will work and be effective.

The implementation view of application requirements presents the real-world manifestation of processing functions and information structures. In some cases, a

physical representation is developed as the first step in application design. However, most computer-based systems are specified in a manner that dictates the accommodation of certain implementation details.

Implementation involves careful planning, investigation of the current system and constraints on implementation, design of methods to achieve the changeover, training of staff in the changeover procedures, and evaluation of changeover methods. The first task is implementation planning i.e. deciding the methods and time scale to be adopted.

Once the planning has been completed, the major effort in the computer department is to ensure that the programs in the system are working properly. At the same time, the user department must concentrate on training user staff. When the staff has been trained, a full system test can be carried out, involving both the computer and clerical procedures.

The main step of implementation includes:

1. Installing user machine.
2. Installing the software on the user's machine.

5.15 System Verification

System Verification is a set of actions used to check the correctness of any element, such as a system element, a system, a document, a service, a task, a requirement, etc. These types of actions are planned and carried out throughout the life cycle of the system. Verification is a generic term that needs to be instantiated within the context it occurs. As a process, verification is a transverse activity to every life cycle stage of the system. In particular, during the development cycle of the system, the verification process is performed in parallel with the system definition and system realization processes and applies to any activity and any product resulting from the activity. The activities of every life cycle process and those of the verification process can work together. For example, the integration process frequently uses the verification process. It is important to remember that verification, while separate from validation, is intended to be performed in conjunction with validation.

The importance of a technology application very likely determines the degree of rigor applied to verifying, testing, and maintaining the technology.

5.16 Alpha Testing

Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or the public. The focus of this testing is to simulate real users by using black-box and white-box techniques. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually, the testers are internal employees of the organization. To put it as simply as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

5.17 Beta Testing

Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered a form of external user acceptance testing. The Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test the product in a real-time environment.

5.18 Types of Beta Testing

There are different types of Beta tests, and they are as follows:

Traditional Beta testing: The product is distributed to the target market, and related data is gathered in all aspects. This data can be used for Product improvement.

Public Beta Testing: The product is publicly released to the outside world via online channels and data can be gathered from anyone. Based on feedback, product improvements can be made.

Table 5.1: Test Case i

Test Case ID: Test001	Test Engineer: Rabia
Objective: Verify that the system is initialized successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Initialize Successfully	
Methods: Initialization Successful	
Comment: Passed	

Table 5.2: Test Case ii

Test Case ID: Test 002	Test Engineer: Rabia
Objective: Verify that the system runs successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Data insert successfully	
Methods: Run the Website Data inserted Successfully	
Comment: Passed	

Table 5.3: Test Case iii

Test Case ID: Test 003	Test Engineer: Rabia
Objective: Verify that the admin/user login successfully Product: Book Nest Environment: Web-Based Application Pre-Requisite: Login displayed successfully	
Methods: Run the Website Admin login successful	
Comment: Passed	

Table 5.4: Test Case iv

Test Case ID: Test 004	Test Engineer: Hira
Objective: Verify that the Product added successfully Product: Book Nest Environment: Web-Based Application Pre-Requisite: Admin login successfully	
Methods: Product added successfully	
Comment: Passed	

Table 5.5: Test Case v

Test Case ID: Test005	Test Engineer: Hira
Objective: Verify that the Product Delete successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Admin login successfully	
Methods: Run the Website	
Open Dashboard	
Product deleted successfully	
Comment: Passed	

Table 5.6: Test Case vi

Test Case ID: Test006	Test Engineer: Rabia
Objective: Verify that the Product Delete successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Admin login successfully	
Methods: Open Dashboard	
Product deleted successfully	
Comment: Passed	

Table 5.7: Test Case vii

Test Case ID: Test005	Test Engineer: Rabia
Objective: Verify that the system initialize successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Initialize successfully	
Methods: Run the Website	
Open Dashboard	
Product deleted successfully	
Comment: Passed	

Table 5.8: Test Case viii

Test Case ID: Test 008	Test Engineer: Rabia
Objective: Verify that the system runs successfully	
Product: Book Nest	
Environment: Android-Based Application	
Methods: Run the Website	
Data inserted Successfully	
Comment: Passed	

Table 5.9: Test Case ix

Test Case ID: Test 009	Test Engineer: Rabia
Objective: Verify that the user login successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Login displayed successfully	
Methods: Run the Website	
User login successful	
Comment: Passed	

Table 5.10: Test Case x

Test Case ID: Test 010	Test Engineer: Rabia
Objective: Verify that that order is placed successfully	
Product: Book Nest	
Environment: Web-Based Application	
Pre-Requisite: Admin login successfully	
Methods: Run the Website	
Order placed successfully	
Comment: Passed	

CHAPTER NO 6 USER MANUAL

6.1 Customer Panel

The **Customer Panel** in the BookNest website is designed to provide users with a simple and user-friendly experience for browsing and inquiring about books. Customers do not need to go through complex checkout processes; instead, they can directly view listings and contact the seller through WhatsApp for more information or to place an order.

Key features of the customer panel include:

- **Book Listing View:** Customers can see all available books with relevant details such as title, author, price, image, and description.
- **Category-Based Browsing:** Books are organized into categories for easier navigation.
- **Search Functionality:** Customers can search for specific books by title or author.
- **WhatsApp Integration:** Each book listing includes a WhatsApp contact button, allowing customers to initiate a conversation directly with the seller.
- **Mobile-Friendly Interface:** The panel is responsive, providing a smooth experience on mobile devices.

This panel ensures a quick and convenient way for users to explore books and make direct inquiries, enhancing overall customer satisfaction.

6.1.1 Main Screen

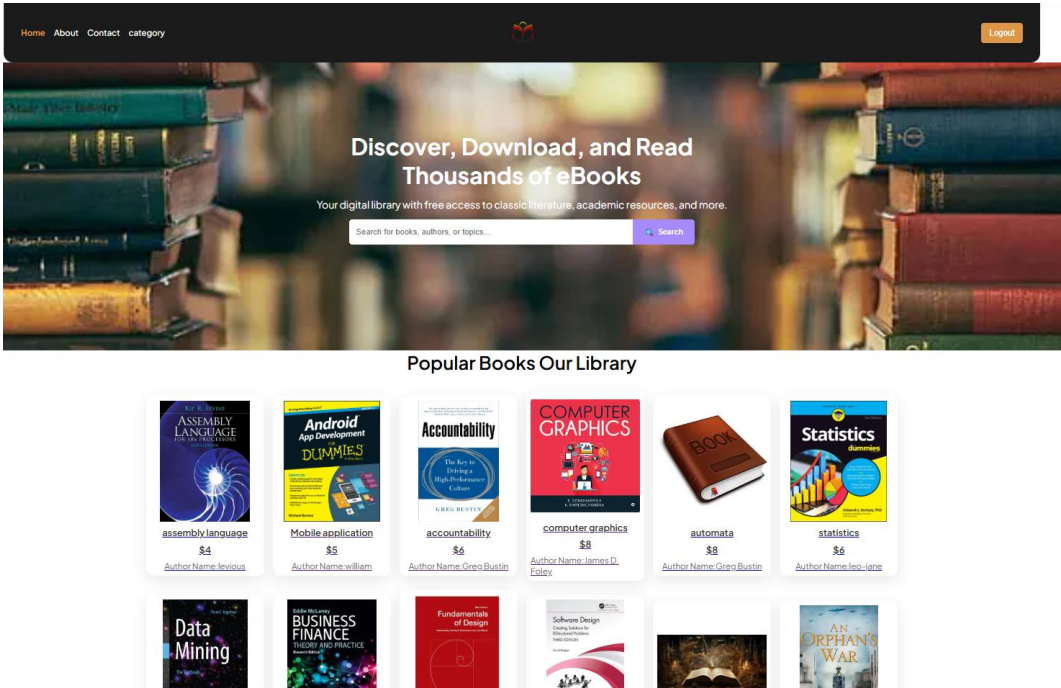


Figure 6.1: Home Screen

6.1.2 Books

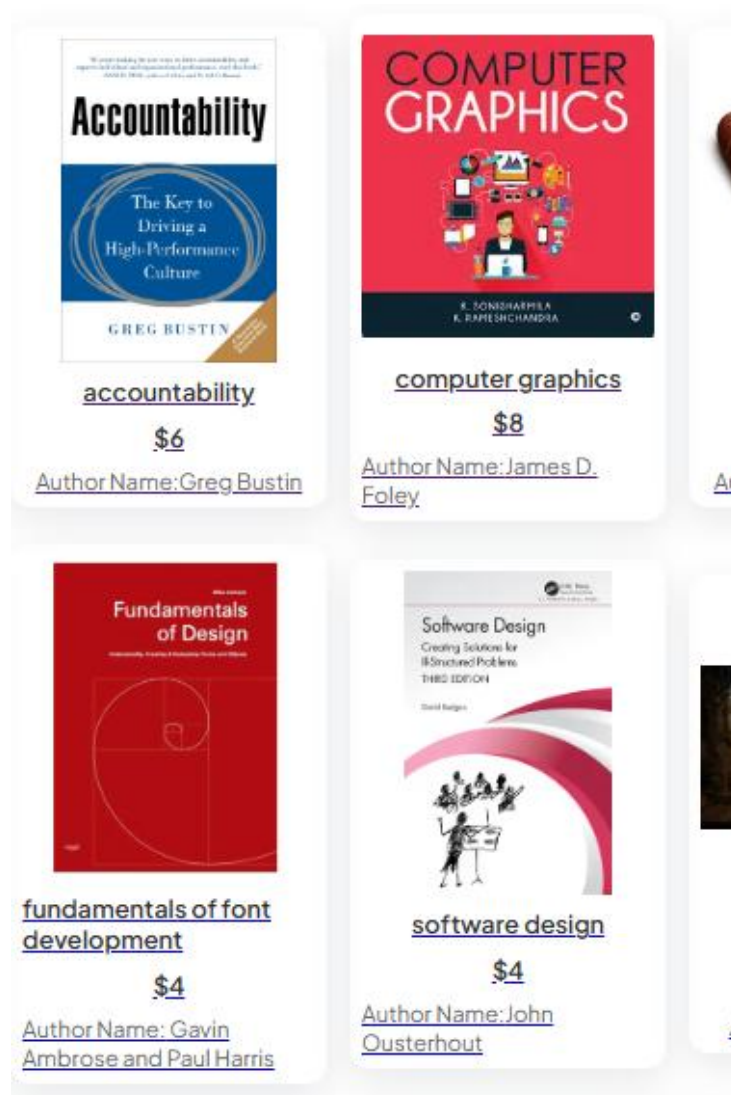


Figure 6.2: Products

6.1.3 Book Detail

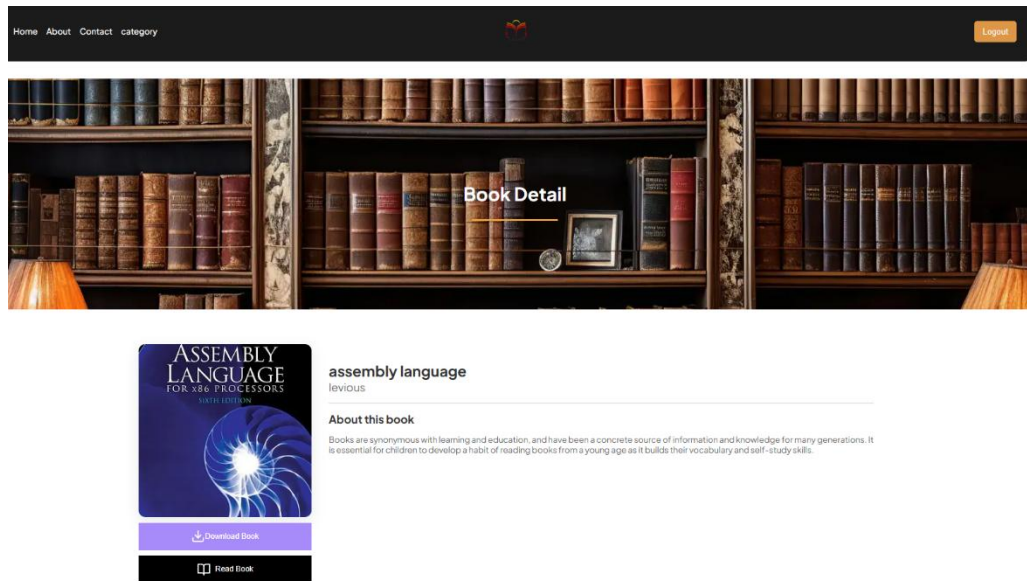


Figure 6.3: Book Detail

6.2 Admin Panel

6.2.1 Dashboard

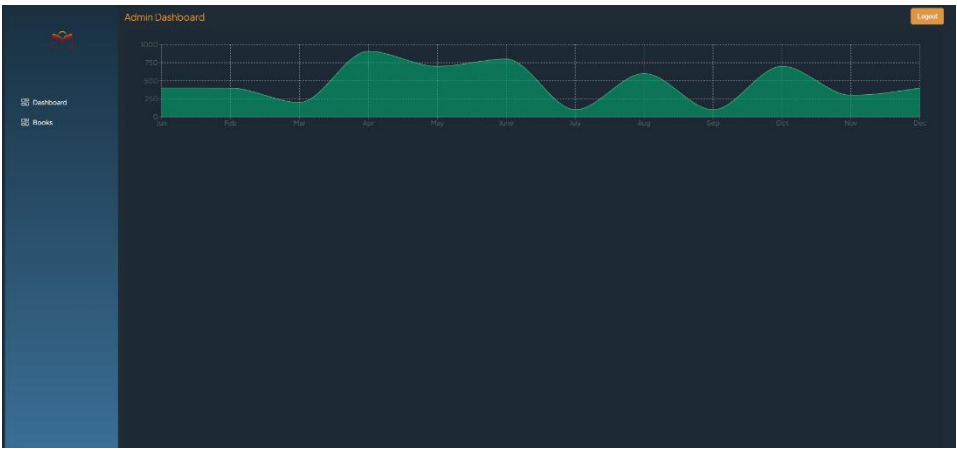


Figure 6.2: dashboard

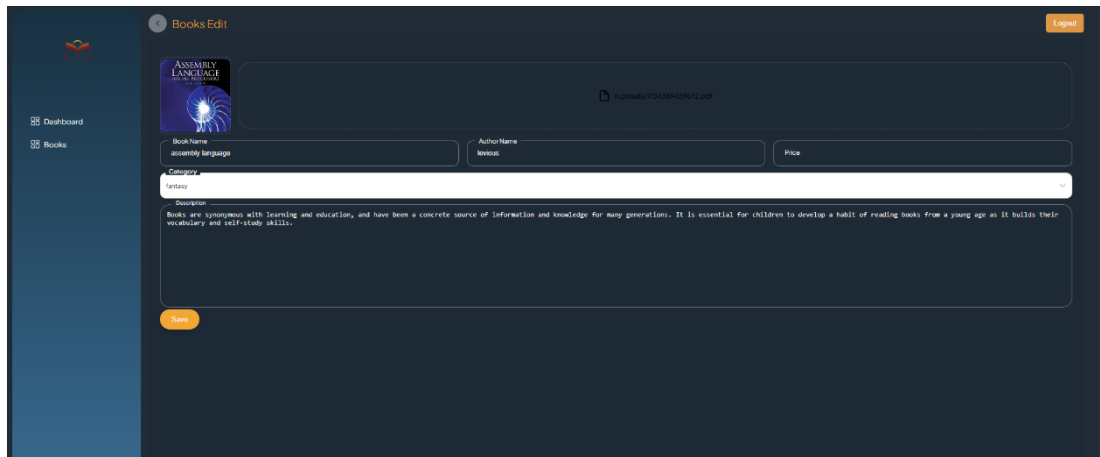
6.2.2 Books

The Books management interface includes a sidebar with 'Dashboard' and 'Books' links. The main area is titled 'Books' and features an 'Add Book' button. Below is a table listing books with columns for Book Name, Picture, PDF, Author Name, Created At, Price, and Action.

Book Name	Picture	PDF	Author Name	Created At	Price	Action
assembly language		Download PDF	Linus	11/07/2023 09:21 PM	\$4	
Mobile application		Download PDF	William	11/07/2023 09:24 PM	\$3	
accountability		Download PDF	Greg Hurst	11/07/2023 09:27 PM	\$4	
computer graphics		Download PDF	James D. Foley	11/07/2023 09:46 PM	\$3	
automata		Download PDF	Greg Hurst	11/07/2023 09:48 PM	\$3	
statistics		Download PDF	Leo-jane	11/07/2023 10:03 PM	\$5	
data mining		Download PDF	Dongsheng Zhang	11/07/2023 10:29 PM	\$4	
business finance		Download PDF	Ross westerfield	23/07/2023 09:59 AM	\$3	
fundamentals of front development		Download PDF	Glen Johnson and Paul Harris	23/07/2023 10:06 AM	\$4	

Figure 6.3: books

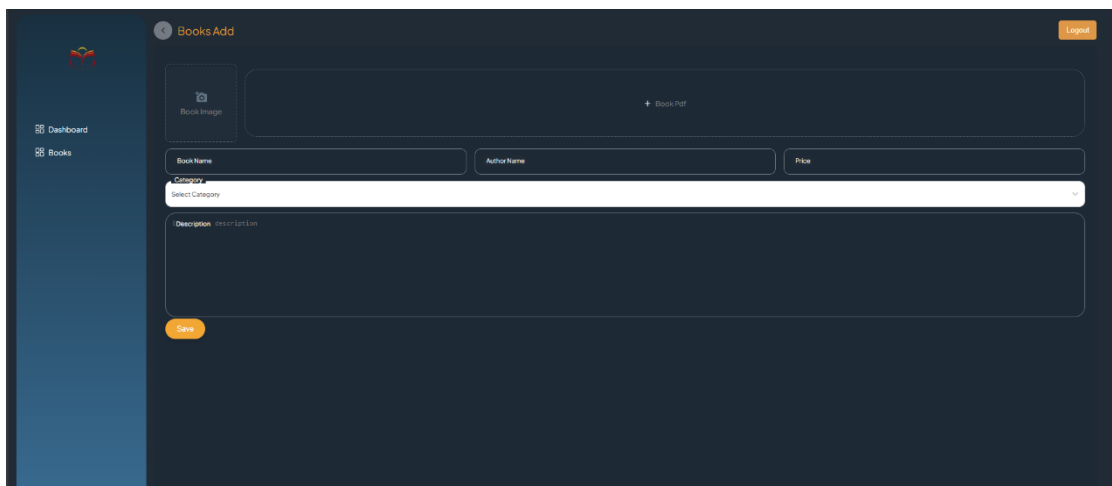
6.2.3 Edit Book



The screenshot shows the 'Books Edit' page. On the left is a sidebar with 'Dashboard' and 'Books' options. The main area has a header 'Books Edit' and a 'Logout' button. Below the header is a book cover image placeholder with a file upload icon and the text 'assemblylanguage1754259429672.pdf'. Underneath are three input fields: 'Book Name' (containing 'assembly language'), 'Author Name' (containing 'benious'), and 'Price'. Below these is a 'Category' dropdown menu showing 'fantasy'. A large text area for 'Description' contains the text: 'Books are synonymous with learning and education, and have been a concrete source of information and knowledge for many generations. It is essential for children to develop a habit of reading books from a young age as it builds their vocabulary and self-study skills.' At the bottom left of the form is a 'Save' button.

Figure 6.4: Edit book

6.2.4 Add Books



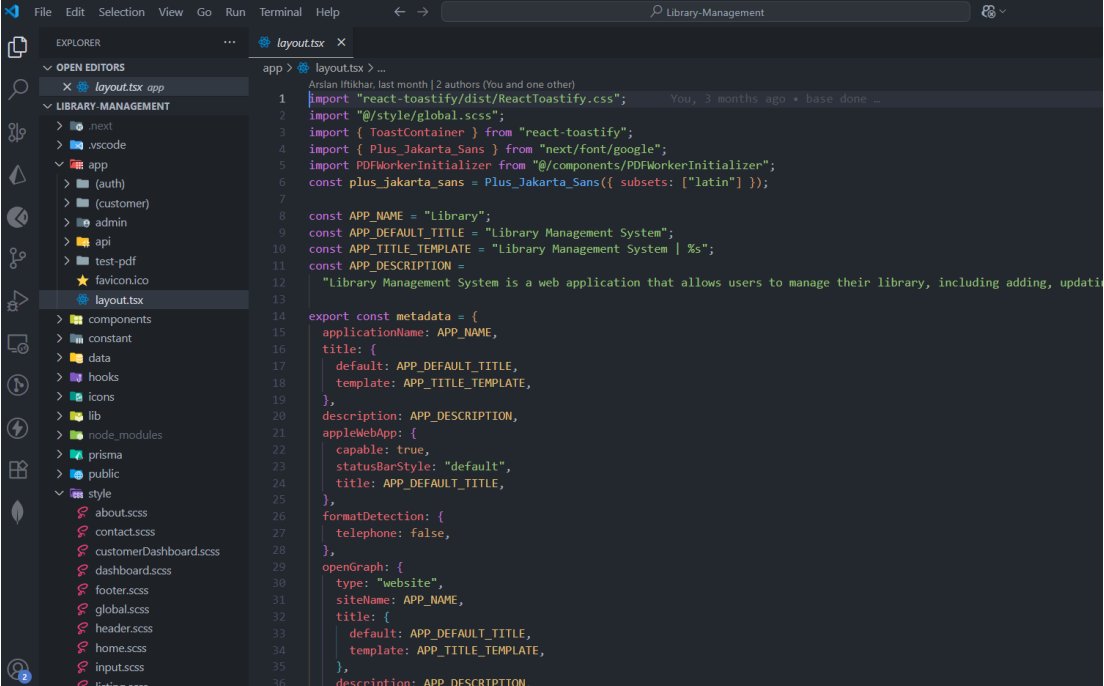
The screenshot shows the 'Books Add' page. On the left is a sidebar with 'Dashboard' and 'Books' options. The main area has a header 'Books Add' and a 'Logout' button. Below the header is a book cover image placeholder with a file upload icon and the text '+ Book.Pdf'. Underneath are three input fields: 'Book Name', 'Author Name', and 'Price'. Below these is a 'Category' dropdown menu showing 'Select Category'. A large text area for 'Description' is empty. At the bottom left of the form is a 'Save' button.

Figure 6.5: Add Product

6.3 Code of Main Activity

The very first activity of the app after the splash screen is the main activity. The front end is built-in React.JS and the backend is in Node. JS.

6.3.1 Main Activity of React App.js file



```
1 import "react-toastify/dist/ReactToastify.css";
2 import "@style/global.scss";
3 import { ToastContainer } from "react-toastify";
4 import { Plus_Jakarta_Sans } from "next/font/google";
5 import PDFWorkerInitializer from "@components/PDFWorkerInitializer";
6 const plus_jakarta_sans = Plus_Jakarta_Sans({ subsets: ["latin"] });
7
8 const APP_NAME = "Library";
9 const APP_DEFAULT_TITLE = "Library Management System";
10 const APP_TITLE_TEMPLATE = "Library Management System | %s";
11 const APP_DESCRIPTION =
12   "Library Management System is a web application that allows users to manage their library, including adding, updating, and deleting books."
13
14 export const metadata = {
15   applicationName: APP_NAME,
16   title: {
17     default: APP_DEFAULT_TITLE,
18     template: APP_TITLE_TEMPLATE,
19   },
20   description: APP_DESCRIPTION,
21   appleWebApp: {
22     capable: true,
23     statusBarStyle: "default",
24     title: APP_DEFAULT_TITLE,
25   },
26   formatDetection: {
27     telephone: false,
28   },
29   openGraph: {
30     type: "website",
31     siteName: APP_NAME,
32     title: {
33       default: APP_DEFAULT_TITLE,
34       template: APP_TITLE_TEMPLATE,
35     },
36     description: APP_DESCRIPTION,
37   },
38 }
```

Figure 6.6: Main code