SOFTWARE DESIGN DOCUMENT

CSE3112: Software Engineering Lab

Title: Boikini

Submitted By:
Shahanaz Sharmin (07)
Anika Tabassum (61)

Contents

1 Introduction

1.1 Purpose

The purpose of this software design document (SDD) is to outline the architecture and system design of Boikini, an online bookshop where customers can buy books, pre-order upcoming releases, and explore a wide range of genres. This document is intended for the development team responsible for building and maintaining the Boikini website. It provides a comprehensive overview of the system's design, components, and functionality, as well as guidance for implementation and testing. By following the guidelines outlined in this document, the development team will be able to create a robust and user-friendly online bookstore that meets the needs and expectations of our customers.

1.2 Scope

Boikini is an online bookshop that provides customers with the ability to browse and purchase a wide range of books, including new releases, bestsellers, and classic titles. In addition to buying books, customers can also pre-order upcoming releases, leave reviews, and create wishlists of books they would like to purchase in the future.

The primary goal of Boikini is to provide a user-friendly and convenient platform for customers to purchase and explore books online. By offering a diverse selection of books, competitive prices, and a seamless checkout process, Boikini aims to attract a wide range of customers, from avid readers to casual book buyers.

Objectives for this project include designing and implementing a responsive and intuitive website interface, integrating secure payment processing and checkout systems, and developing an efficient inventory management system to ensure that books are accurately tracked and restocked as necessary.

The benefits of Boikini for both customers and the development team are significant. For customers, Boikini provides a convenient and accessible way to purchase books from the comfort of their own home, with the added benefit of pre-ordering upcoming releases and leaving reviews to help inform other customers' purchasing decisions. For the development team, Boikini represents an opportunity to build a robust and scalable e-commerce platform that can be expanded and customized over time to meet the evolving needs of customers.

1.3 Overview

1.3.1 Introduction

This software design document (SDD) provides a comprehensive overview of the architecture and system design of Boikini, an online bookshop. This document is intended for the development team responsible for implementing the system and stakeholders who wish to understand the system's design.

1.3.2 System Description

The SDD begins with a high-level overview of the system architecture, including a description of the various components and their interactions. This is followed by detailed descriptions of each component, including the web server, database, payment processing system, and inventory management system.

1.3.3 Design Overview

This section provides a summary of the design rationale for selecting the system architecture described in this document. It outlines the key design considerations and trade-offs that were made in order to achieve the desired functionality, scalability, and maintainability.

1.3.4 Object Model

The object model section provides an overview of the system's object-oriented design, including a description of the various objects and their relationships. It also includes class diagrams and other visual aids to illustrate the system's object model.

1.3.5 Subsystem Decomposition

This section describes how the system is decomposed into subsystems, including a description of the responsibilities of each subsystem and the interactions between them.

1.3.6 Data Design

The data design section describes the system's data model and the various data structures and relationships that exist within the system. It also includes entity relationship diagrams and other visual aids to illustrate the system's data design.

1.3.7 User Requirements and Component Traceability Matrix

The user requirements and component traceability matrix section provides a detailed description of the system's user requirements and how they map to individual components within the system. It includes a matrix that shows how each user requirement is satisfied by one or more system components.

1.4 Reference Material

The following documents were used as sources of information for the development of Boikini:

- The project requirements document, which outlines the functional and non-functional requirements of the system
- The project proposal, which provides a high-level overview of the project goals and objectives
- https://dev.mysql.com/doc/
- https://developer.mozilla.org/en-US/docs/Glossary/MVC

These documents were essential in guiding the development of the Boikini website, and they were referenced frequently throughout the design and implementation process. By drawing on the information and insights contained in these documents, the development team was able to build a robust and user-friendly online bookshop that meets the needs and expectations of customers.

1.5 Definitions and Acronyms

The following terms, acronyms, and abbreviations are used throughout this software design document:

- **Boikini**: The name of the online bookshop being developed.
- **SDD**: Software design document.
- **UI**: User interface, which refers to the visual design and layout of the website.
- **API**: Application programming interface, which provides a set of functions and protocols for accessing a web-based software application.
- **SQL**: Structured Query Language, which is a programming language used for managing and querying relational databases.
- **HTTP**: Hypertext Transfer Protocol, which is a protocol used for transmitting data over the internet.
- **HTTPS**: Hypertext Transfer Protocol Secure, which is a secure version of HTTP that encrypts data transmitted over the internet.
- **SSL**: Secure Sockets Layer, which is a security protocol used for establishing encrypted connections between web servers and clients.

- **PCI DSS**: Payment Card Industry Data Security Standard, which is a set of security standards developed by major credit card companies to ensure the secure handling of credit card information.

These definitions and acronyms are provided to help ensure that all readers of this document can properly interpret and understand its contents. If any additional terms or acronyms are introduced throughout the document, they will be defined in context.

2 System Description

Boikini is an online bookshop that allows users to browse, purchase, and pre-order books from a wide selection of titles. The website is designed to be user-friendly, with a clean and intuitive interface that allows users to easily find the books they are looking for. The system is designed to handle a large volume of transactions, and includes secure payment processing capabilities to ensure the safe and reliable handling of customer data.

The functionality of Boikini is divided into two main areas: browsing and purchasing. Users can browse the book catalog by searching for specific titles or authors, or by browsing through categories such as fiction, non-fiction, and children's books. Once they have found a book they are interested in, they can view details such as the title, author, cover image, and description, as well as user reviews and ratings.

If the book is available for purchase, the user can add it to their cart and proceed to checkout. At this point, they will be prompted to enter their shipping and billing information, and to choose a payment method. Boikini supports a variety of payment methods, including credit card and various mobile finanical services, and uses secure encryption protocols to ensure the safe transmission of sensitive information.

For books that are not yet available for purchase, users can choose to pre-order them. The website will provide an estimated release date, and will automatically charge the user's payment method and ship the book as soon as it becomes available.

In terms of context, Boikini is intended to be used by anyone who enjoys reading books, whether for personal or professional reasons. The website is accessible from any internet-connected device, and is designed to be easy to use even for those with limited technical expertise.

The design of the Boikini system is based on a client-server architecture, with a web-based user interface that communicates with a back-end server and database. The website is built using modern web development frameworks and tools, including HTML, CSS, JavaScript, and Python. The server-side logic is implemented using the Django web framework, which provides a robust and flexible platform for building web-based applications. The database is implemented using MySQL, which provides a fast and reliable data storage solution that can handle a large volume of transactions. Finally, the system includes integration with third-party payment processing APIs, which provide secure and reliable payment processing capabilities.

3 Design Overview

3.1 Design Rationale

The architecture for Boikini's online bookshop was chosen after careful consideration of several critical issues and trade-offs. One of the most important considerations was scalability - the ability of the system to handle a large volume of users and transactions. To achieve this, we chose a modular architecture that allows for easy scaling of individual components as needed.

Another key consideration was security. As an online bookstore, Boikini handles sensitive user data such as payment information and personal details. To ensure the security of this data, we implemented a client-server architecture that ensures more security and easier maintenance.

We also considered various alternative architectures before settling on the chosen one. One alternative was a layered architecture, where there are multiple layers of components that are placed into logical groupings based on the type of functionality they provide or based on their interactions with other components. While this would have been easier to implement, it would have limited our ability to scale as the framework does not allow for growth. It would also make the maintenance difficult.

We also thought about a microservices architecture, where functionality is broken down into small, independent services. While this architecture would have allowed for greater flexibility and scalability, it would have also introduced additional complexity and overhead, and may not have been necessary for the size and scope of our project.

Ultimately, we believe that the chosen architecture strikes the right balance between scalability, security, and maintainability, and provides a solid foundation for the development of a robust and user-friendly online bookshop.

3.2 System Architecture

The Boikini online bookshop system is designed to be modular, with each subsystem responsible for a specific set of functions. The high-level subsystems and their assigned roles or responsibilities are as follows:

- **Web Server Subsystem:** This subsystem is responsible for managing the user interface and user interactions with the system. It handles all user requests, manages user sessions, and provides access to the various other subsystems.
- **Database Subsystem:** This subsystem is responsible for storing and retrieving all data used by the system, including user account information, book metadata, transaction records, and inventory levels.
- **Payment Processing Subsystem:** This subsystem is responsible for handling all financial transactions, including securely processing customer payments and managing refunds or chargebacks as necessary.

- **Inventory Management Subsystem:** This subsystem is responsible for managing the inventory of books available for purchase on the site. It tracks inventory levels, updates them as new orders are placed, and alerts staff when inventory levels fall too low.

These subsystems work together to provide the full functionality of the Boikini system. The Web Server Subsystem acts as the primary interface for users, directing requests to the appropriate subsystem and relaying responses back to the user. The Database Subsystem provides the necessary data to support all other subsystems, while the Payment Processing and Inventory Management Subsystems handle the specific tasks related to financial transactions and inventory management.

3.3 Constraint and Assumptions

3.3.1 List of Assumption

The following assumptions have been made in the development of the Boikini online bookshop:

1. Customers have a valid and functioning internet connection to access the website.
2. Customers have a valid and functioning payment method to purchase books.
3. All book information and data are accurate and up-to-date.
4. The inventory management system will be able to handle all book orders and stock levels accurately.
5. The payment processing system will be secure and able to handle all financial transactions.
6. The web server hosting the website will have sufficient resources and capacity to handle all incoming traffic and requests.
7. The development team has the necessary expertise and resources to implement all planned features and functionality.
8. The website will comply with all relevant laws and regulations related to online book sales and data privacy.

3.3.2 List of Dependencies

These assumptions were made based on research and industry standards, and may need to be revisited and updated as the development of the project progresses. The successful development and deployment of the Boikini online bookshop system depends on the availability and proper functioning of the following software and hardware components:

- Web server software

- Database management system
- Payment processing gateway
- E-commerce platform
- Programming languages and frameworks
- Operating system
- Network infrastructure
- Internet connectivity and bandwidth
- Web browser software
- Mobile device operating systems

It is assumed that all of these dependencies will be available and properly configured during the development and deployment phases of the project. Any changes or updates to these dependencies should be carefully tested and validated to ensure that they do not adversely affect the functionality or stability of the Boikini online bookshop system.

4 Object Model

4.1 Object Description

Class name	User
Brief Description	Defines a user
Attributes	1. userName : string 2. password : string 3. email : string 4. shipping address : string 5. payment information : string
Methods with Minimal Description	1. createUser() : Creates a user, takes email password and other fields as parameters. Returns if created successfully. 2. login(): Logins a user if the user inputs correct credentials. Denies access otherwise. 3. forgotPassword(): Shows reset password page if the user forgets their current password. 4. editProfile(): Shows the edit profile page 5. getUserData(): Show user's data

Class name	Books
Brief Description	Defines a Book
Attributes	1. title: string 2. author : Date 3. publisher : string 4. price : int 5. type : string
Methods with Minimal Description	1. addBook() : Addss a Book, takes different fields as parameters. Returns if created successfully. 2. editBook(): Edit a specific book 3. deleteBook(): Delete a specific book

Class name	Review
Brief Description	Defines a review
Attributes	1. userName: string 2. Date : Date 3. rating : string 4. description : string 5. status : string
Methods with Minimal Description	1. giveReview() : Submits a new review for a book. 2. editReview(): Edits a review given before 3. deleteReview(): Deletes a review

Class name	Order
Brief Description	Defines an order
Attributes	1. orderNumber: string 2. placedDate : Date 3. customerId : string 4. description : string 5. listOfBooks : string
Methods with Minimal Description	1. placeOrder() : Places an order, takes different fields as parameters. Returns if created successfully. 2. confirmOrder(): Confirms a specific order 3. cancelOrder(): Deletes a specific order

Class name	Cart
Brief Description	Describes a cart details
Attributes	1. cartSession: string 2. Date : Date 3. Book : string 4. quantity : Integer
Methods with Minimal Description	1. addToCart() : Add a new item to cart 2. UpdateCart(): Edits a cart item 3. deleteFromCart(): Deletes a item from the cart

Class name	Payment
Brief Description	Defines a payment
Attributes	1. orderNumber: string 2. placedDate : Date 3. customerId : string 4. description : string 5. listOfBooks : string
Methods with Minimal Description	1. paymentType() : Defines which way to pay, takes different fields as parameters. Returns if created successfully. 2. transactionID(): saves the information about the customer with their trasaction ID. 3. paymentmethod(): Defines payment method. 4. confirmPayment() : Confirms a payment

Class name	Notification
Brief Description	Represents a notification sent to a user
Attributes	1. userName: string 2. Date : Date 3. content : string 4. status : string
Methods with Minimal Description	1. sendNotification() : Sends a notification to the user. 2. markasRead(): Marks the notification as Read. 3. deleteNotification(): Deletes a notification

4.2 Object Collaboration Diagram

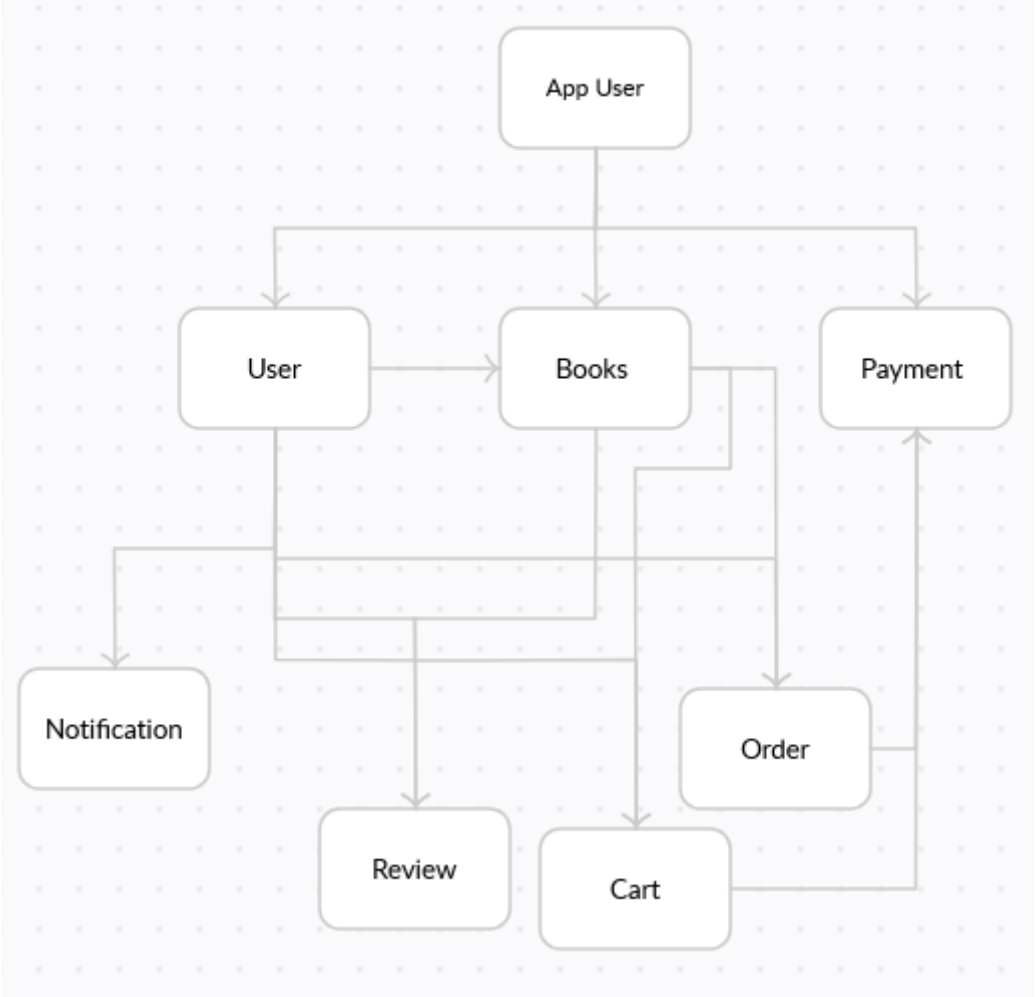

Figure 1: Object Collaboration Diagram

5 Subsystem Decomposition

5.1 Complete Package Diagram

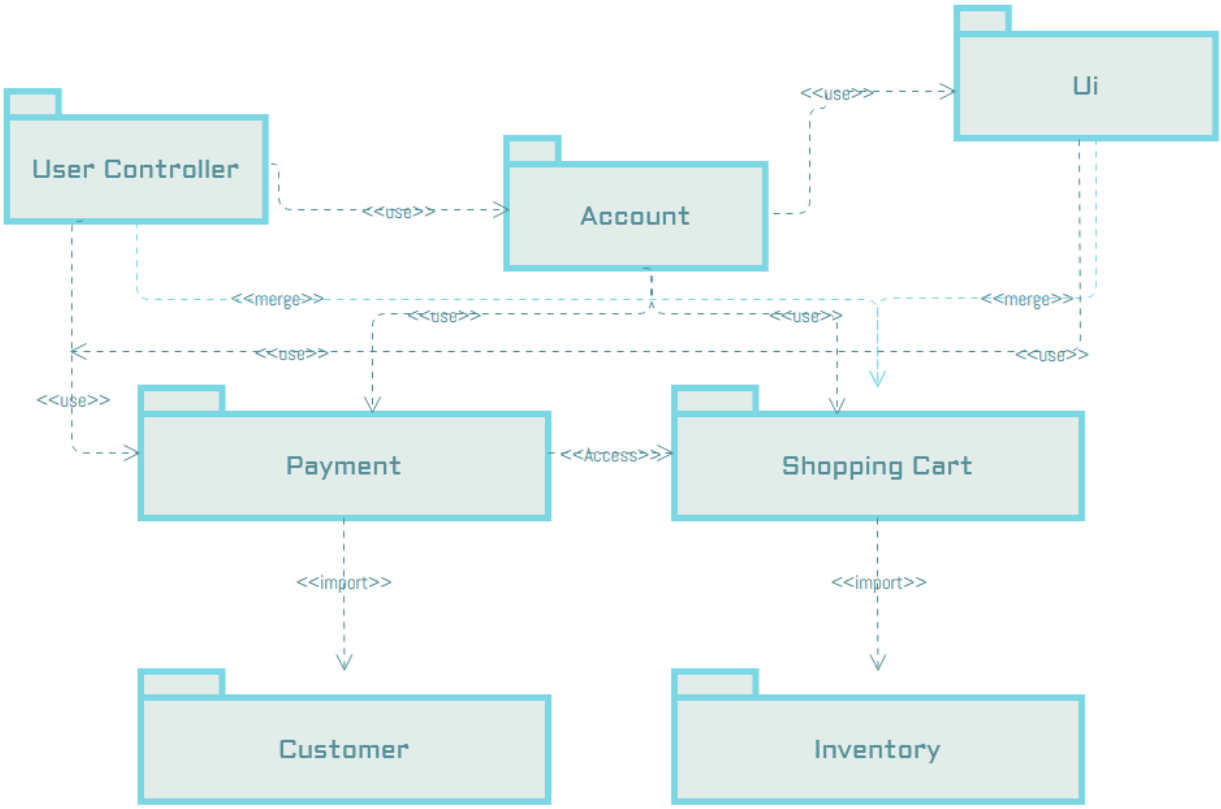

Figure 2: Complete Package Diagram

5.2 Subsystem Detail Description

5.2.1 UserController

5.2.1.1 Module Description

This controller controls all the interactions of the user

5.2.1.2 Class Diagram

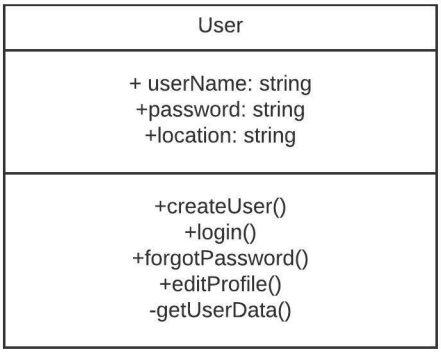

Figure 3: UserController

5.2.1.3 Subsystem Interface

The user will first be presented with a login or signup page. The user will be able to create a new account if it doesn't exist or login with proper credentials. The user will see the main page after logging in.

5.2.2 Account

5.2.2.1 Module Description

This controller controls all the interactions of the user accounts

5.2.2.2 Class Diagram

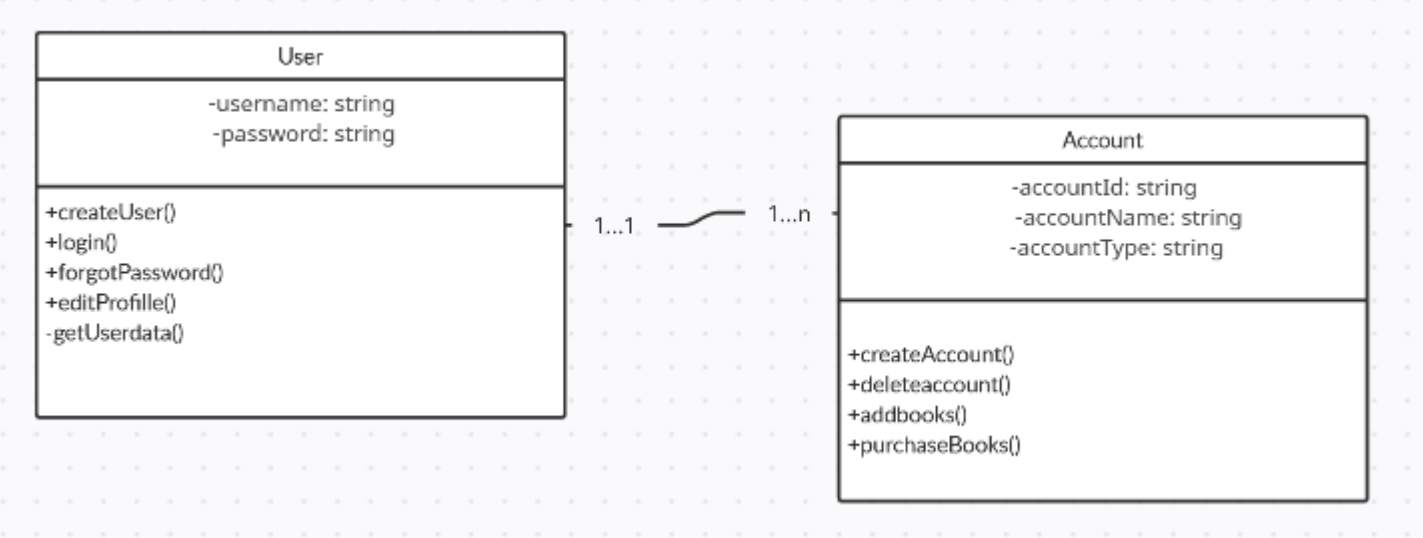

Figure 4: AccountController

5.2.2.3 Subsystem Interface

The user will see the main page from here. The user will be able to add new account to search for books, buy books or remove books from the list.

5.2.3 Payment

5.2.3.1 Module Description

This controller controls all the interactions of the payments

5.2.3.2 Class Diagram

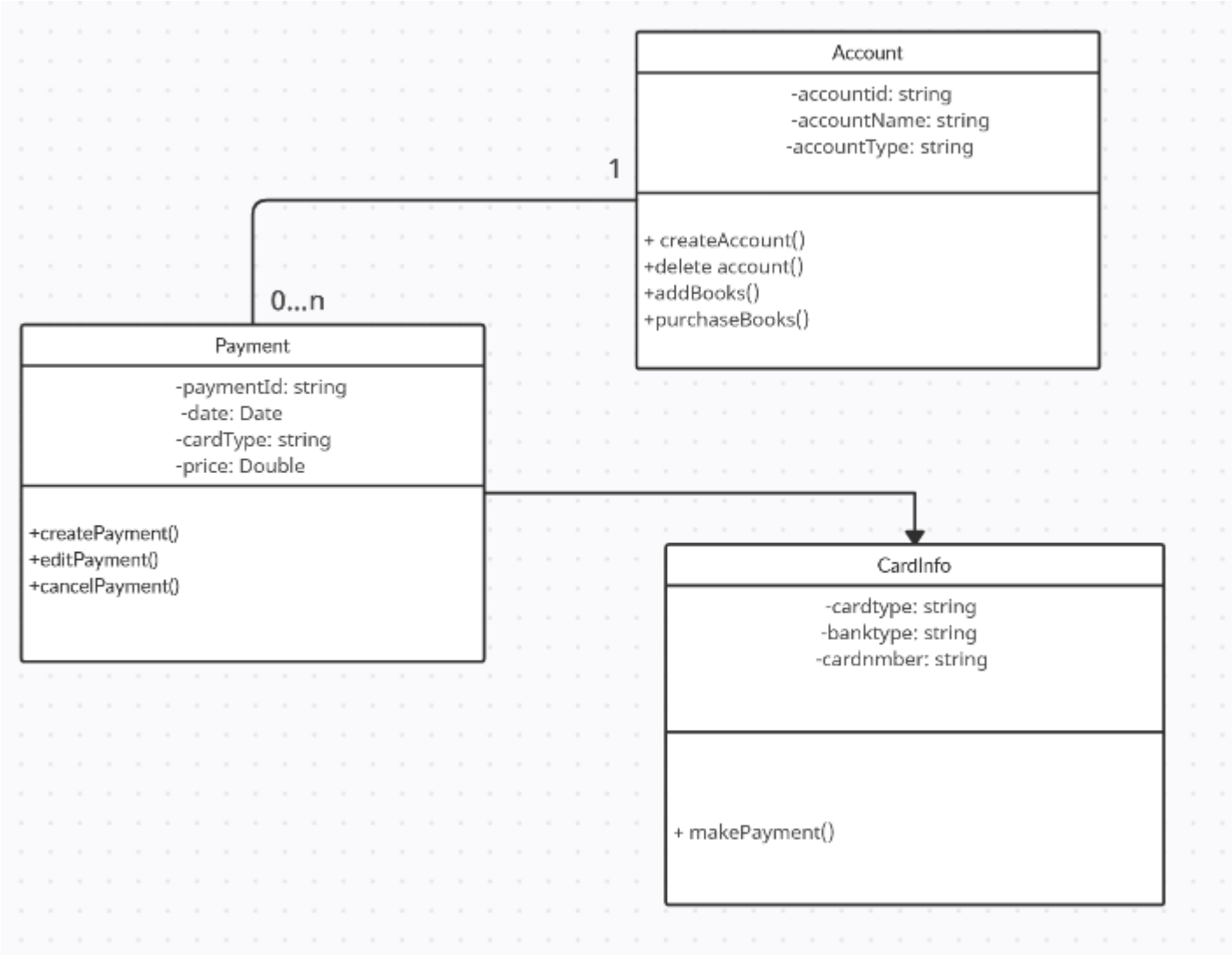

Figure 5: Payment

5.2.3.3 Subsystem Interface

The payment package is a subsystem of the Boikini project responsible for handling all payment-related functionalities. The payment package is designed to interact with other subsystems of the project.

5.2.4 Shopping Cart

5.2.4.1 Module Description

This controller controls all the interactions of the shopping carts and purchases

5.2.4.2 Class Diagram

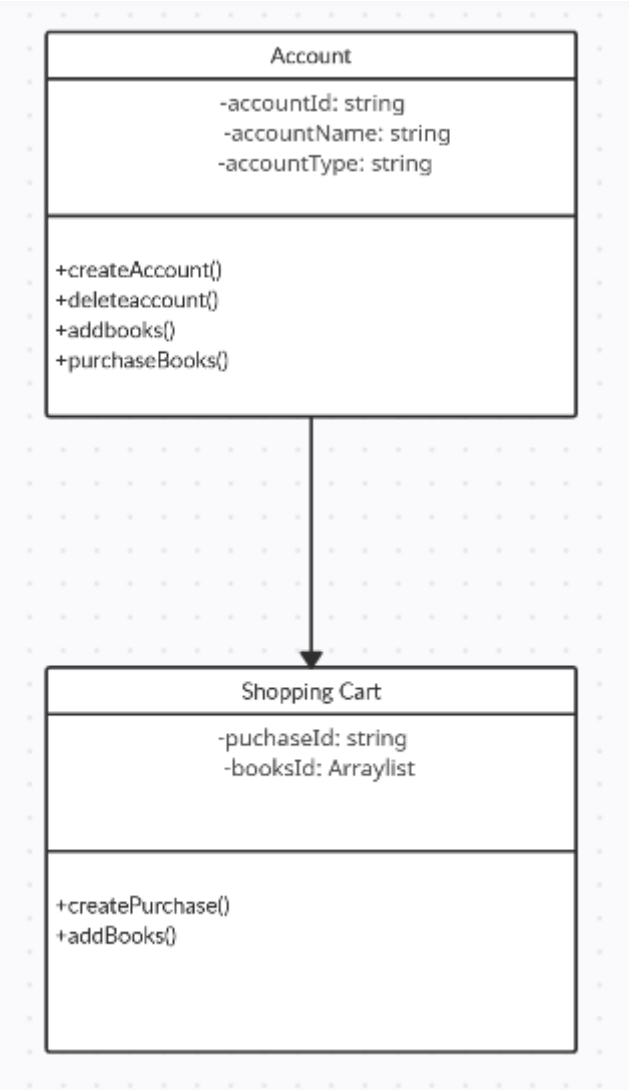

Figure 6: Shopping cart

5.2.4.3 Subsystem Interface

The shopping cart package is responsible for managing the items that customers have added to their cart, calculating the total cost of the items, and enabling customers to checkout their cart.

5.2.5 Customer

5.2.5.1 Module Description

This controller controls all the interactions of the customers and about the purchase of the books

5.2.5.2 Class Diagram

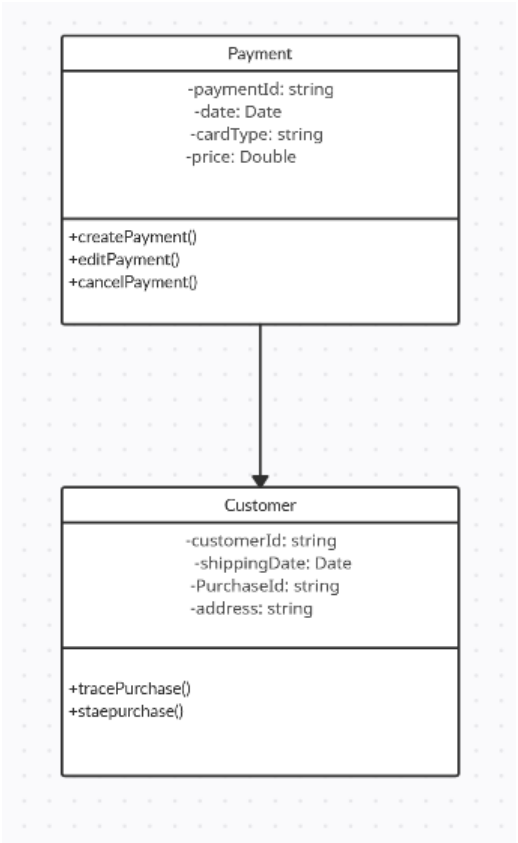

Figure 7: Customer

5.2.5.3 Subsystem Interface

The customer package is responsible for handling customer-related functionality, including tracing the purchase, to know where the packages are etc.

5.2.6 Inventory

5.2.6.1 Module Description

This controller controls all the interactions of the raw materials used to produce goods as well as the goods that are available for sale.

5.2.6.2 Class Diagram

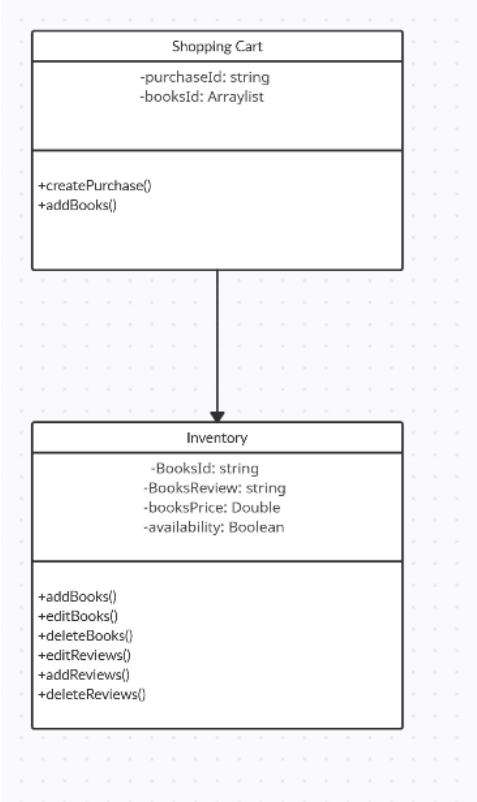

Figure 8: Inventory

5.2.6.3 Subsystem Interface

The Inventory Package is responsible for managing the inventory of the Boikini bookstore. It provides functionality for adding new books to the inventory, updating existing books, and retrieving book information.

6 Data Design

6.1 Data Description

Our system Boikini typically consists of various types of information such as book details, user information, order history, payment details, and more. To transform this information domain into data structures, the system needs to organize and store data in a structured format that can be easily accessed and manipulated.

One way to represent this information domain is through the use of data models. Data models provide a conceptual representation of the data, which can then be translated into data structures for implementation. There are several data models that can be used, but one commonly used model is the entity-relationship (ER) model.

The major data or system entities in an online bookstore system can be stored in a database or data storage items. These entities may include:

- **Book entity**: This entity stores book details such as book title, author name, ISBN, publisher, and price.
- **User entity**: This entity stores user details such as username, password, email address, shipping and billing addresses, and payment details.
- **Order entity**: This entity stores order details such as the order number, order date, book details, and shipping details.
- **Payment entity**: This entity stores payment details such as payment method, card number, card holder name, and billing address.
- **Review entity**: This entity stores book reviews submitted by users, including the book title, reviewer name, rating, and comments.

To store and process these entities, a database management system (DBMS) will be used. A DBMS is software that manages the storage, retrieval, and manipulation of data in a database.

The data in the database is typically organized into tables, where each table represents an entity. Each row in the table represents a specific instance of the entity, and each column represents an attribute of the entity. For example, the book entity table will have columns for book ID, title, author, publisher, and price.

The data is processed using SQL (Structured Query Language), which is a programming language used to communicate with databases. SQL can be used to perform various operations on the data, such as adding, deleting, or updating records.

In summary, the information domain of Boikini system is transformed into data structures through the use of data models, which are then stored in a database or data storage items using a DBMS. The data is organized into tables and processed using SQL.

6.2 Data Dictionary

The following is an alphabetical list of the major data entities in the system, along with their types and descriptions :

- **Book** :
  - **Attributes:**
    - title (string): the title of the book
    - author (string): the author of the book
    - publisher (string): the publisher of the book
    - ISBN (string): the International Standard Book Number of the book
    - price (float): the price of the book
    - quantity_available (integer): the number of copies of the book available for purchase
  - **Methods:**
    - get_title(): returns the title of the book
    - set_title(title: string): sets the title of the book to the given value
    - get_author(): returns the author of the book
    - set_author(author: string): sets the author of the book to the given value
    - get_publisher(): returns the publisher of the book
    - set_publisher(publisher: string): sets the publisher of the book to the given value
    - get_ISBN(): returns the ISBN of the book
    - set_ISBN(ISBN: string): sets the ISBN of the book to the given value
    - get_price(): returns the price of the book
    - set_price(price: float): sets the price of the book to the given value
    - get_quantity_available(): returns the quantity of the book available for purchase

    * set_quantity_available(quantity_available: integer): sets the quantity of the book available for purchase to the given value

- **Cart**:
  - **Attributes:**
    * items (list): a list of book objects in the cart
    * total_price (float): the total price of all items in the cart
  - **Methods:**
    * add_item(item: Book): adds a book object to the cart
    * remove_item(item: Book): removes a book object from the cart
    * get_total_price(): returns the total price of all items in the cart
- **Order**:
  - **Attributes :**
    * order_number (string): the unique order number
    * customer (User): the customer who placed the order
    * items (list): a list of book objects in the order
    * order_date (datetime): the date and time the order was placed
    * shipping_info (string): the shipping information of the order
    * payment_info (string): the payment information of the order
  - **Methods :**
    * get_order_number(): returns the order number
    * get_customer(): returns the customer who placed the order
    * set_customer(customer: Customer): sets the customer who placed the order to the given value
    * get_items(): returns the list of book objects in the order
    * add_item(item: Book): adds a book object to the order
    * remove_item(item: Book): removes a book object from the order
    * get_order_date(): returns the date and time the order was placed
    * get_shipping_info(): returns the shipping information of the order
    * set_shipping_info(shipping_info: string) : sets the shipping info
- **User**:
  - **Attributes :**
    * userId (string) : the Id of the User
    * name (string): the name of the user

  - email (string): the email address of the user
  - shipping_address (string): the shipping address of the user
  - billing_address (string): the billing address of the user
  - payment_information (Payment object): the payment information of the user

- **Methods :**
  - get_name(): returns the name of the user
  - set_name(name: string): sets the name of the user to the given value
  - get_email(): returns the email address of the user
  - set_email(email: string): sets the email address of the user to the given value
  - get_shipping_address(): returns the shipping address of the user
  - set_shipping_address(shipping_address: string): sets the shipping address of the user to the given value
  - get_billing_address(): returns the billing address of the user
  - set_billing_address(billing_address: string): sets the billing address of the user to the given value
  - get_payment_information(): returns the Payment object containing the payment information of the customer

6.3 Entity Relationship Diagram

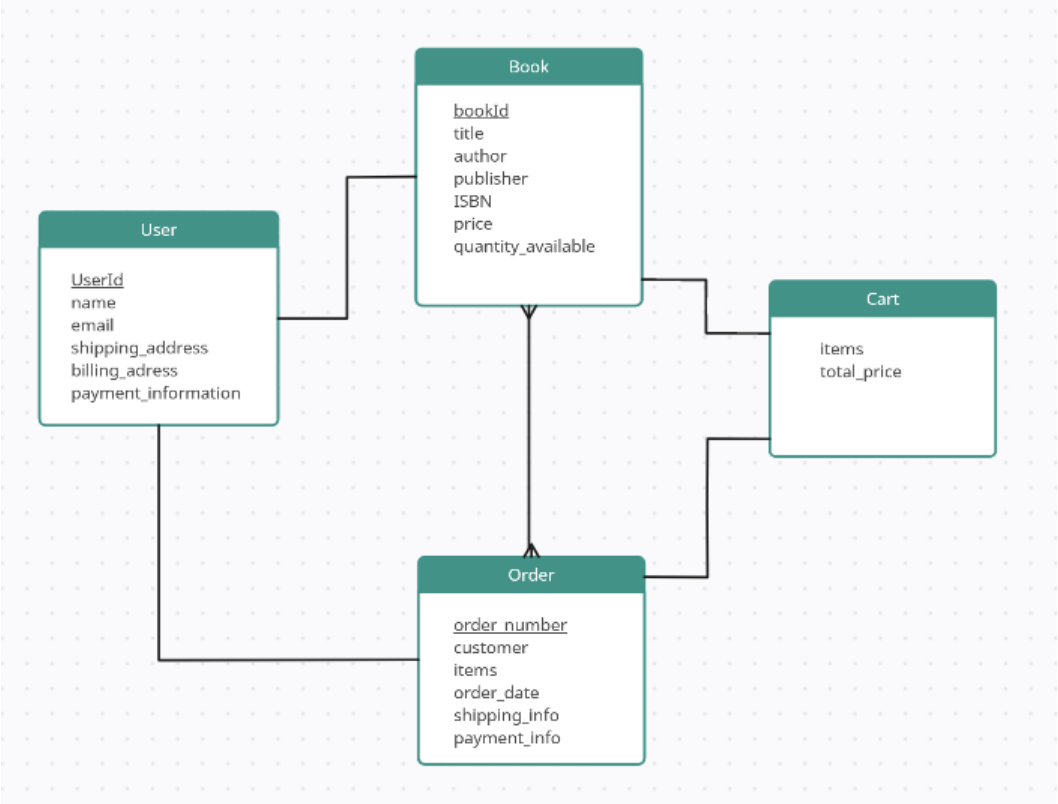

Figure 9: ER Diagram

7 User Requirement and Component Traceability Matrix

User Requirement	Component
User Authentication	User, Authentication, PasswordRecovery, Session
Search Function	SearchEngine, SearchResult, Service
User Profile Management	User,UserManager
Payment Gateway	Payment, PaymentMethod, PaymentGateway, PaymentAuthorization
Notification System	Notification
Review	Review, ReviewManager
Reporting Function	ReportingManager,Feedback
Contacting	Message,Inbox
Order Status	Order, OrderManager
Order and Customer Information	Order,User