Group 14

BOOKEE
Software Architecture Document

Version 2.0

Revision History

Date	Version	Description	Author
29/11/2023	1.0	First version with the information we have at sprint 3	PTPNam LTHThu TTNgan
17/12/2023	2.0	Update Logical view & the View component Add Deployment & Implementation view	PTPNam LTHThu TTNgan

Table of Contents

Software Architecture Document

1. Introduction

This article includes an overview of the "Bookee System" as well as explanations of how it operates. It will define how a user views and interacts with the system, such as searching books, adding products to shopping cart and editing profile, etc. This also describes the web's architecture, the use cases supported by the system, as well as the architectural styles and materials required to obtain the use cases.

1.1 Purpose

This document provides a comprehensive architectural overview of the Bookee System, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document applies to the whole Bookee System, addressing the background for this project, and the architecturally significant functional requirements. Each of the aforementioned quality attributes will be described through a comprehensive set of scenarios followed by an architectural overview, which includes a bird's eye view and a full description of patterns and tactics that will be used to address the core quality attributes. This will be followed up by a look at a couple views into the system.

1.3 Definitions, Acronyms and Abbreviations

Term list:

- Stakeholder: any person involved or affected, directly or indirectly, by this product.
- JavaScript: web-browser interpreted programming language for enhancing websites in a dynamic way.
- Bootstrap: a free frontend framework for faster and easier web development, includes HTML and CSS, as well as optional JS plugins.

Acronym list:

- SAD: Software Architecture Document
- MVC: Model-View-Controller model, a software architecture pattern that separates the physical way to store data, the business logic and the appearance to the user.
- UI: User Interface
- API: Application Programming Interface, a protocol used as an interface to allow communication between different components.
- REST: Representational State Transfer, web API featuring a stateless client-server infrastructure.

1.4 References

None.

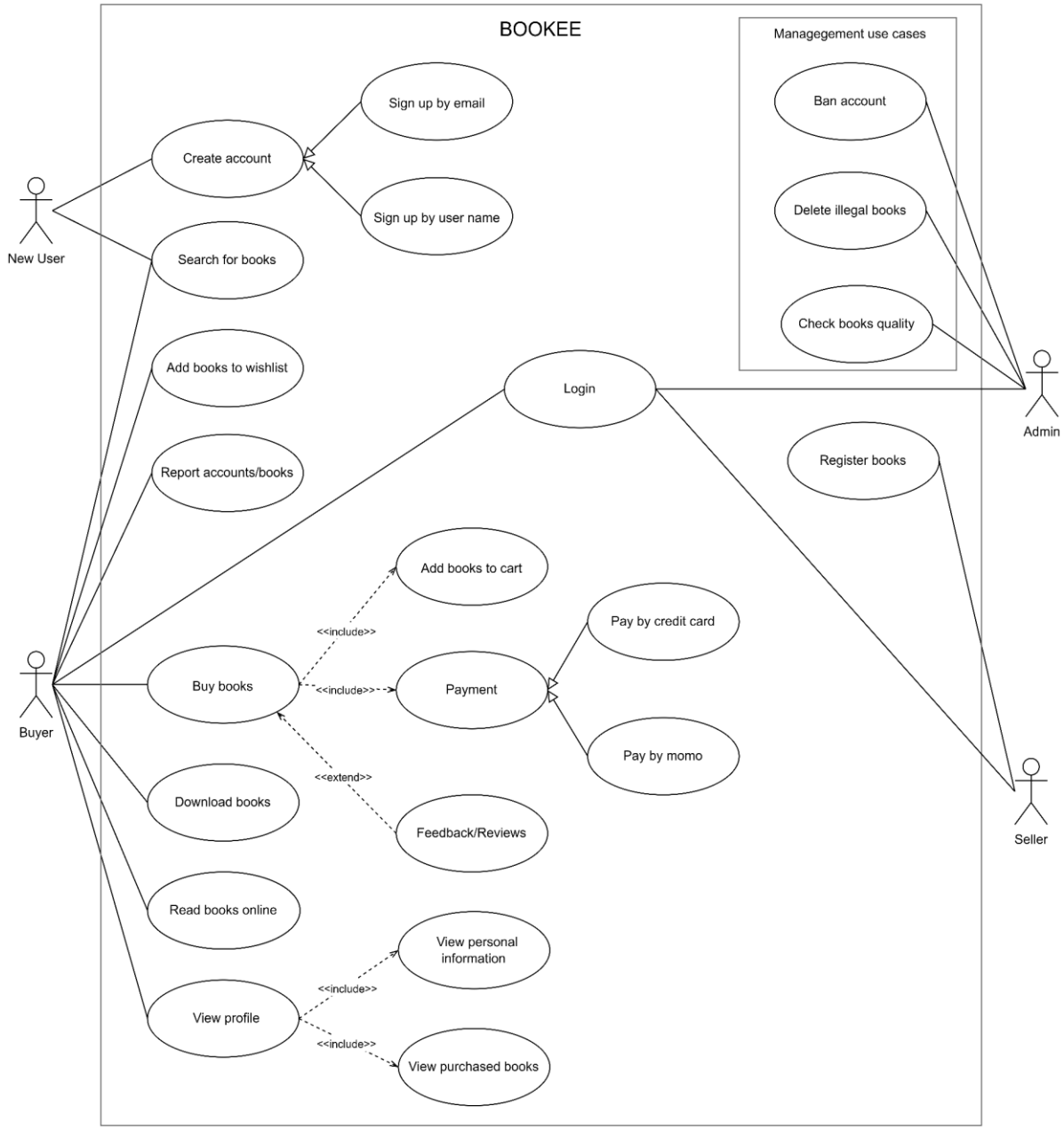
1.5 Overview

The architectural design of the Bookee is detailed in the next section. First, the system's fundamental software architecture will be established. Then there is more debate on the goals and limits that will be put on the quality of the finished product, such as security, distribution, and reuse. The primary perspectives of the scheme are shown in the preceding sections to represent various features of the system. Finally, criteria for system size, performance, and quality will be suggested.

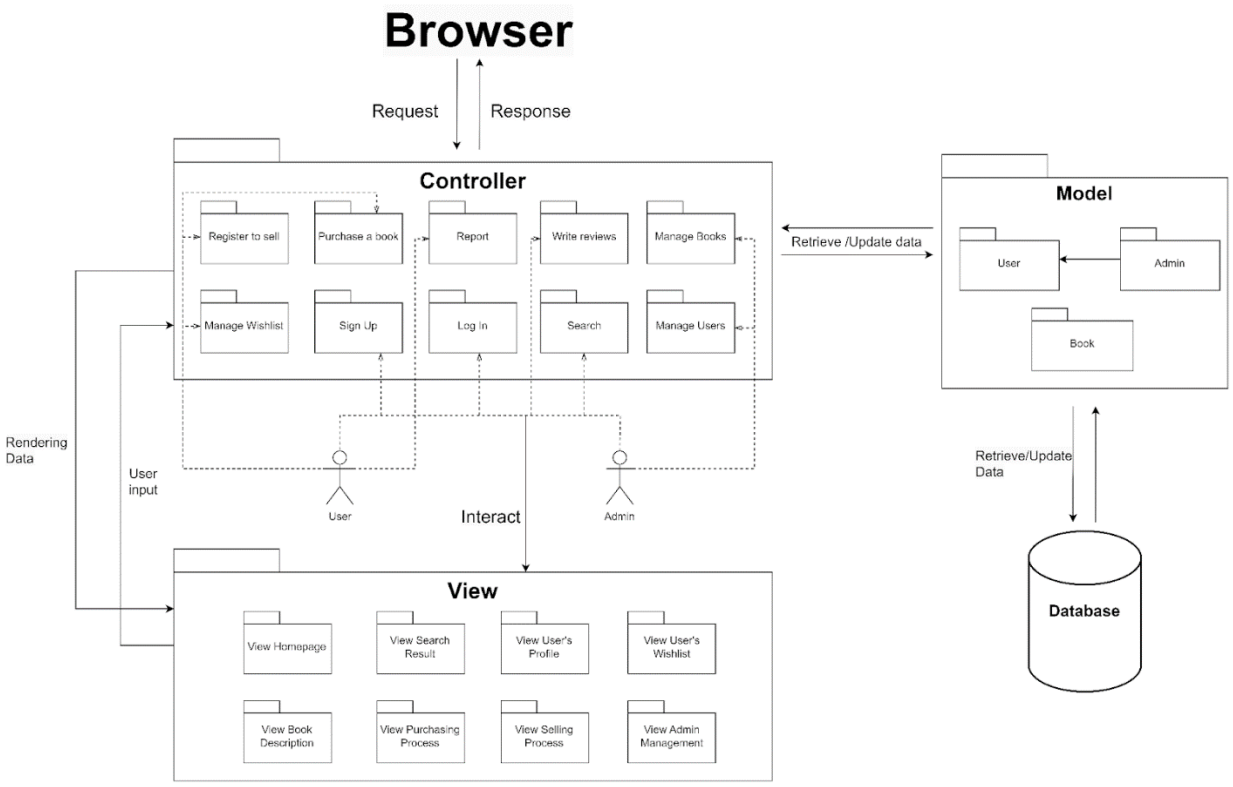
2. Architectural Goals and Constraints

- **Programming language:** JavaScript, CSS, HTML *(markup language)*.
- **Database management system:** PostgreSQL.
- **Environment:** Web.
- **Security:** information about the user's account (especially username and password), credit card or transactions are secured and encrypted.
- **Performance:** each request is expected to be responded rapidly or in acceptable time.

3. Use-Case Model



4. Logical View

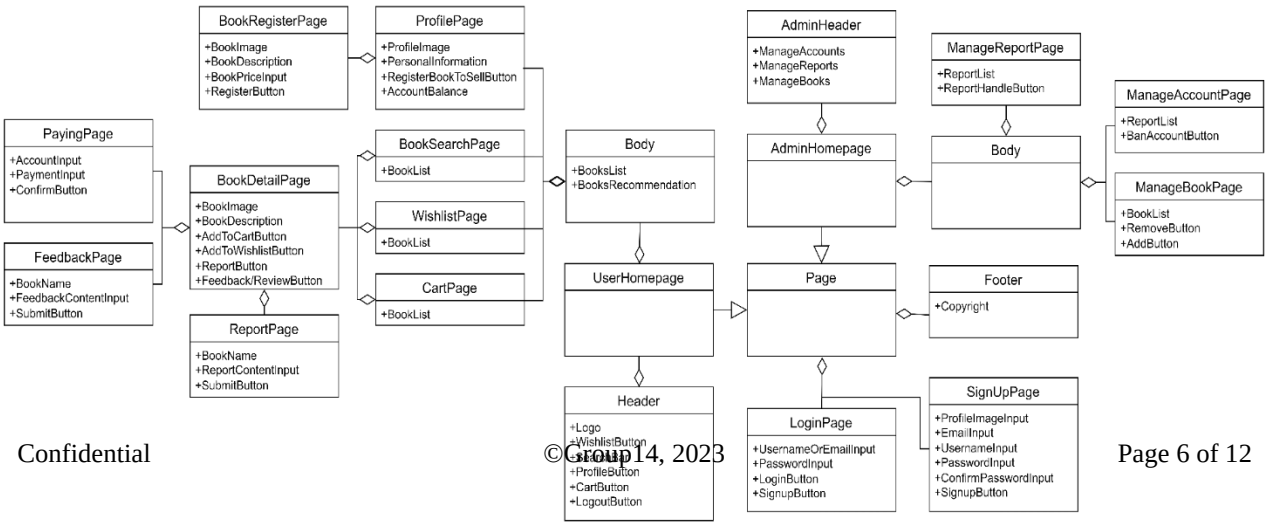


The architecture pattern used for this system is a MVC (Model-View-Controller) pattern. MVC separates an application into three interconnected components:

- Model - contains the core functionality and data
- View - displays the information to the user
- Controller - handles the input from the user

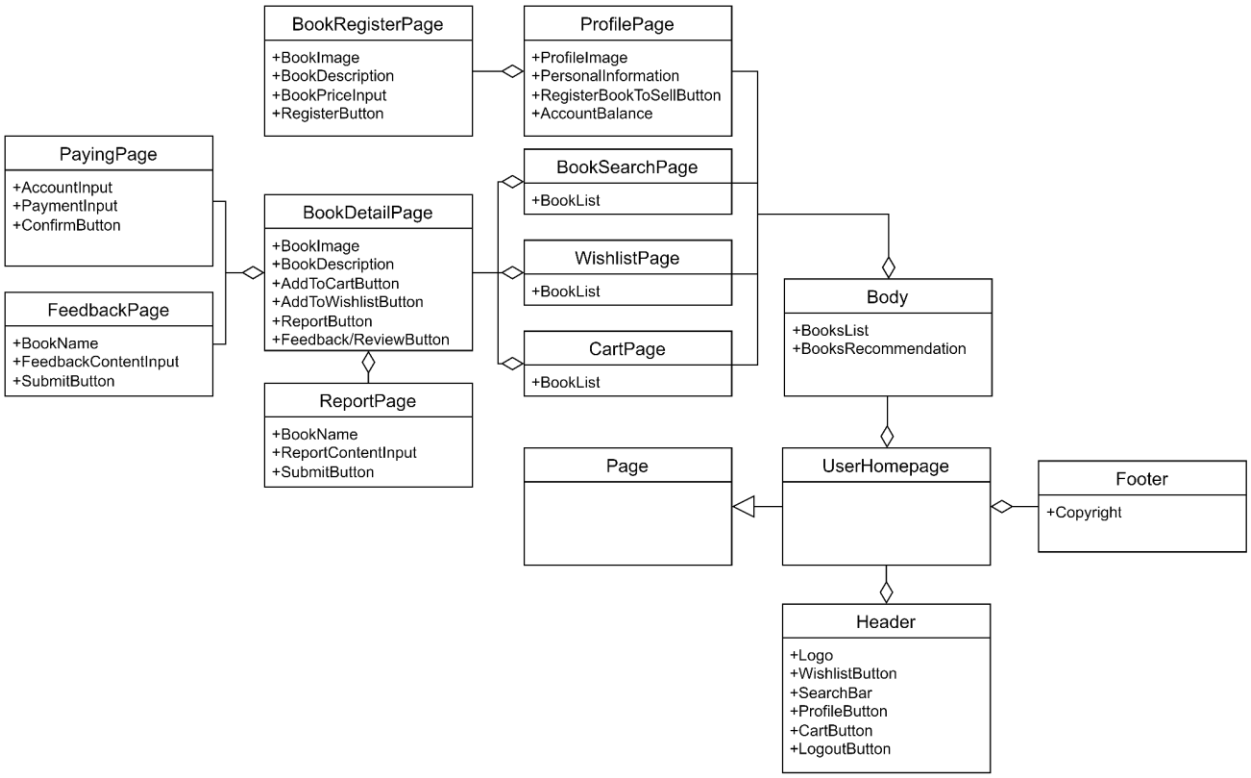
Each component has a distinct role and responsibility, contributing to the overall structure and organization of the code.

4.1 Component: View



In the MVC model, the view presents the application's user interface (UI) to users. The View interacts with the Model indirectly through the Controller. It can request data from the Model and update its display based on the changes in the Model. Each page consists of three sections: the Header, Body, and Footer. Users can interact with various web components, which are then sent to the controller for further processing.

4.1.1 User View

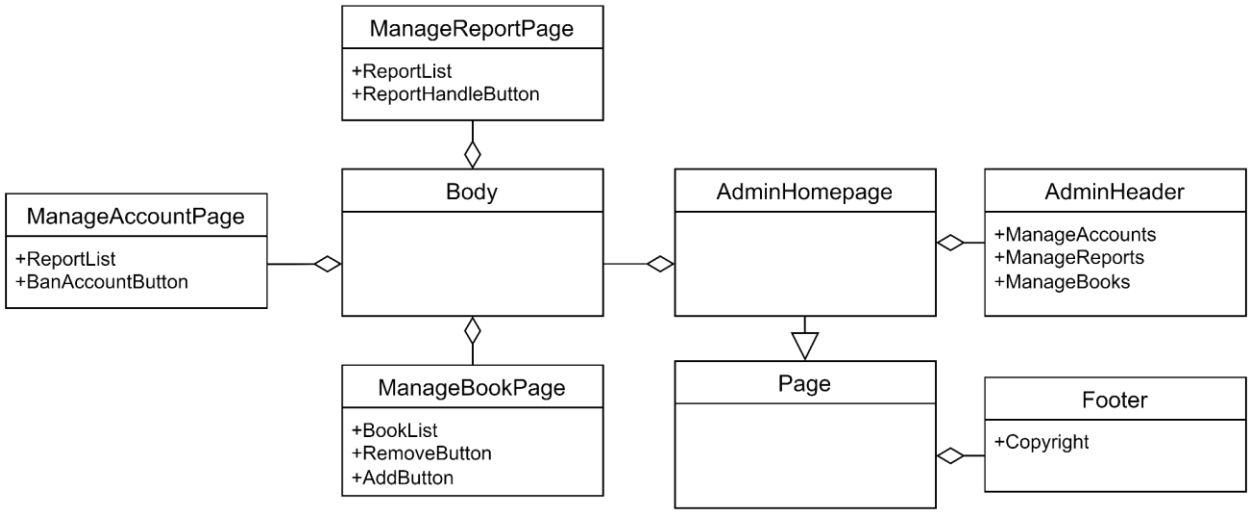


The diagram above illustrates the user's interface when using the web platform. Upon logging in and accessing the home page, a comprehensive list of web books will be displayed. Users can conveniently navigate through various functions using the header, which includes the following options:

1. ProfilePage: This page showcases the user's personal information.
2. BookSearchPage: Users can view their book search results on this page.
3. WishlistPage: Here, users can find the books they have selected and added to their wishlist.
4. CartPage: This page presents a list of books that the user intends to purchase.

When a user clicks on a specific book, they will be directed to the BookDetailPage. On this page, users can access detailed information about the book. Additionally, users have the option to report the book on the report page, provide feedback about the book, or proceed with purchasing the book. If the user chooses to buy the book, they will be redirected to the PayingPage, where they can securely fill in the necessary information.

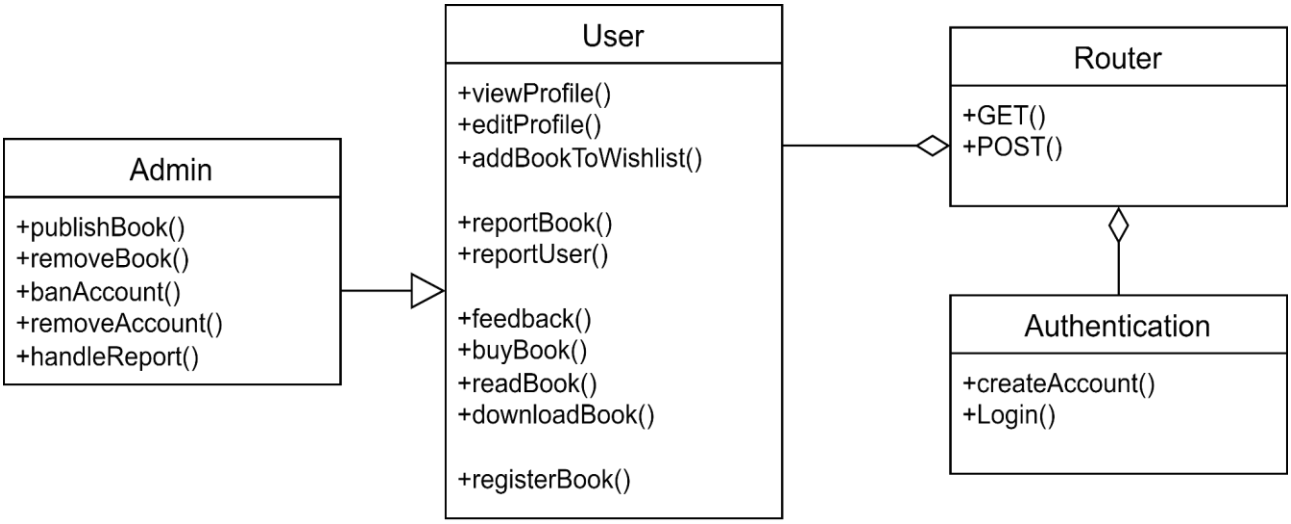
4.1.2 Admin View



As an admin, once logged in, the admin has the ability to manage data through the web platform. There are specific pages designed for administrative tasks, including:

1. ManageReportPage: This page provides a comprehensive overview of user reports regarding books. The admin can review these reports and take appropriate actions, such as deleting books from the database if necessary.
2. ManageAccountPage: On this page, you can access reports filed by users against other users. The admin has the authority to review these reports and handle them accordingly, which may include actions like banning the reported user's account.
3. ManageBookPage: This page displays a list of books that users have registered to sell. The admin can oversee and manage this inventory, ensuring its accuracy and compliance with platform guidelines.

4.2 Component: Controller



In the MVC model, controllers serve as the intermediary between the model and view components, facilitating request processing, data manipulation using the model, and interaction with the view to generate the final output. **The Router** directs user requests to the appropriate controller functions. These functions retrieve the necessary data from the model, render an HTML page, and return it for the user to view in the browser.

User Controller: handle all the interactions and inputs from the user

- viewProfile(): Displays the user's personal profile.
- editProfile(): Allows the user to modify their personal profile.
- addBookToWishlist(): Adds a book to the user's wishlist.
- reportBook(): Reports a book as inappropriate or illegal.
- reportUser(): Reports another user.
- feedback(): Enables the user to leave feedback or a review for a purchased book.
- buyBook(): Facilitates the purchase of a book.
- readBook(): Opens a purchased book for the user to read.
- downloadBook(): Allows the user to download a purchased book.
- registerBook(): Registers a book to be sold to the admin.

Admin Controller: handle all the interactions and inputs from the admin

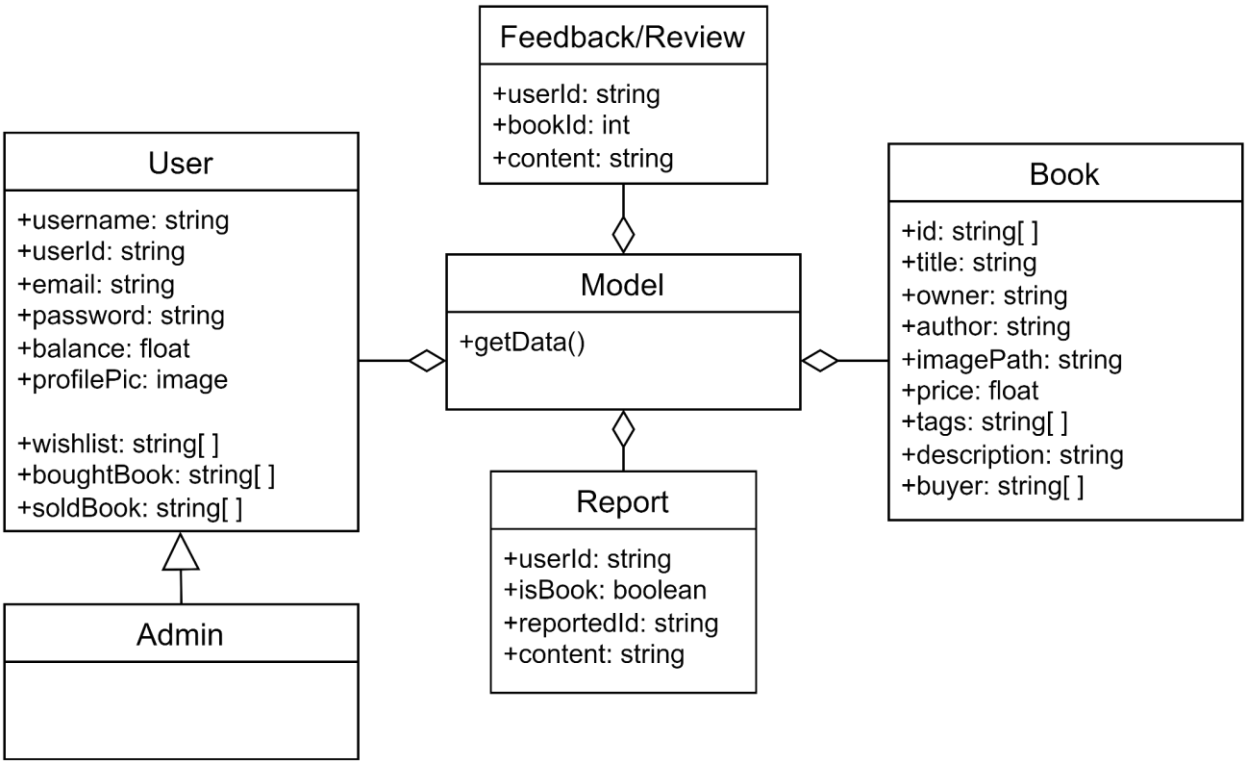
- publishBook(): Accepts a book registration from a user and adds it to the database.
- removeBook(): Removes a book from the database and the web.
- banAccount(): Bans an inappropriate or illegal user account.
- removeAccount(): Removes an account from the database.
- handleReport(): Manages all user reports.

Authentication Controller: call to the general services corresponding to the authentication process

- createAccount(): Creates a user account and adds it to the database.
- login(): Enables user login to the web application.

These controllers handle various interactions and inputs from users and admins, ensuring smooth functionality and proper data management within the application.

4.3 Component: Model



The Model component connects to the database to load and store data under control of the Controller. It has the following important parts:

- **User** Model: contains various attributes to help distinguish between users
  + username: a string variable that represents the username of a user.
  + userId: a string variable that represents the unique identifier of a user.
  + email: a string variable that represents the email address of a user.
  + password: a string variable that represents the password when a user creates an account.
  + balance: a float variable that represents the account balance of a user, which will be used to purchase books.
  + profilePicture: an image that will be displayed when the user's account is active.
  + wishlist: an array of string variables that stores the id of books in the wishlist of user.
  + boughtBook: an array of string variables that stores the id of books that a user purchased.
  + soldBook: an array of string variables that stores the id of books that are sold by a user.

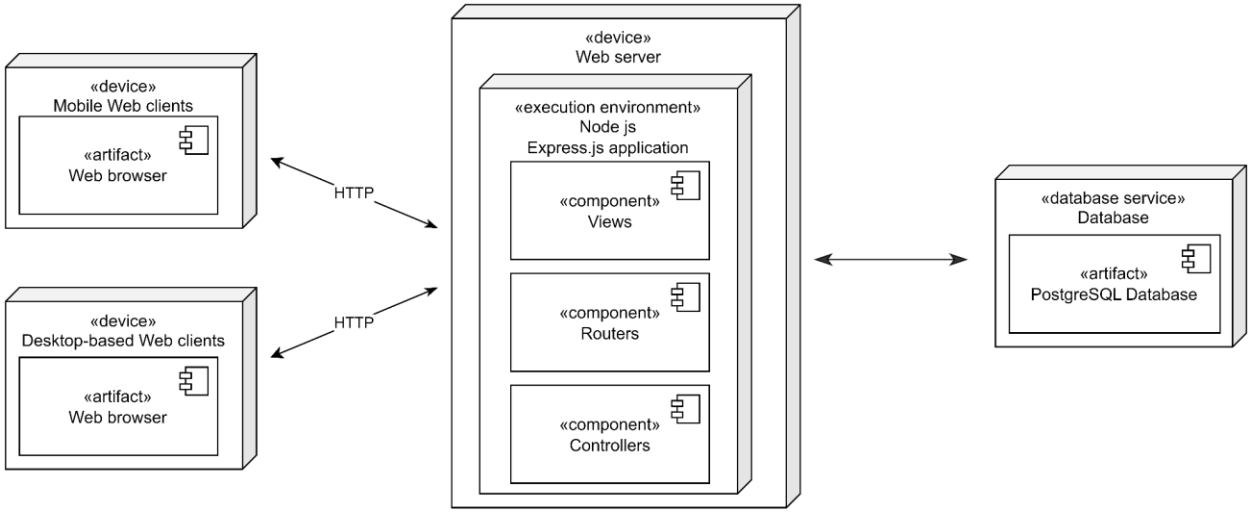
- **Book** Model: contains various attributes to help distinguish between books
  + id: a string variable that represents the identifier of a book.
  + title: a string variable that represents the book title.
  + owner: a string variable that represents the name of the owner whose book belongs to.
  + author: a string variable that represents the name of the author of the book.
  + imagePath: a string variable that stores the hypertext reference to the cover of the book.
  + price: a float variable that represents the price of the book.
  + tags: an array of string variables that contains different genres that the books belong to.

+ description: a string variable that represents the brief introduction of the new book.
+ buyers: an array of string variables that stores the id of buyers who bought the book.

- **Report** Model: contains various attributes to collect data from the reports user send to admin
  + userId: a string variable that stores the identifier of the user that reports.
  + isBook: a boolean variable that marks if the report is towards the book or another user.
  + reportedId: a string variable that stores the identifier of the reported book/user.
  + content: a string variable.

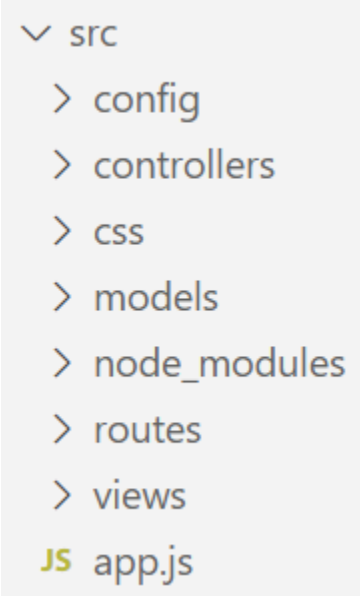
- **Feedback/Review** Model: contains various attributes to collect data to display opinion of user on the book
  + userId: a string variable that stores the identifier of the user that writes the feedback.
  + bookId: a string variable that stores the identifier of the book that receives feedback.
  + content: a string that represents the opinion on the book.

5. Deployment



Users access the app by launching a web browser and entering the app's URL. The app's server receives the user's request through the HTTP protocol and proceeds to process it. The request is then routed to the appropriate controller by the router, which determines the necessary actions to be executed. The controller effectively processes the data retrieved from the database and generates a response, which is subsequently sent back to the user.

6. Implementation View



- The config folder contains the module responsible for connecting to the database.

- The controllers folder holds the controllers that handle user requirements and interactions.

- The css folder stores stylesheets that define the visual appearance of the web pages.

- The models folder contains the database schema, which defines the structure and relationships of the data.

- The node_modules folder includes JavaScript functions and libraries used by the application.

- The routes folder consists of routers that handle user requests and direct them to the appropriate controllers.

- The views folder contains HTML code that renders the user interface, displaying the app's functionality and content.