ALT+F4

Read & Relay
Software Architecture Document

Version 1.1

Revision History

Date	Version	Description	Author
30/11/23	1.0	First version of the Software Architecture Document	Entire team
13/12/23	1.1	Supplied content for sections 5 and 6	Entire team

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This Software Architecture Document (referred to as SAD from here on) provides a comprehensive architectural overview of the Read & Relay book-selling platform by illustrating different aspects of the system using various architectural views, with the aim of capturing and communicating the significant architectural decisions made on the system.

1.2 Scope

This SAD provides a high-level overview of the architectural design of Read & Relay. It includes a summary of the main components and their interactions, as well as the fundamental design principles of the system.

1.3 References

- Sommerville, I. (2011) *Software Engineering*. 9th Edition. Addison-Wesley.
- Pressman, R.S. (2001) *Software Engineering: A Practitioner's Approach*. 5th Edition. McGraw Hill.
- Lecture reference materials provided by the course's lecturers.

1.4 Definitions and acronyms

- Read & Relay: The system whose architecture this document concerns. Read & Relay is a used book trading & selling platform where each user is both a buyer and a seller.
- SAD: Software Architecture Document.
- MCV: Model - View - Controller, a software design pattern.
- UI: User Interface.

2. Architectural Goals and Constraints

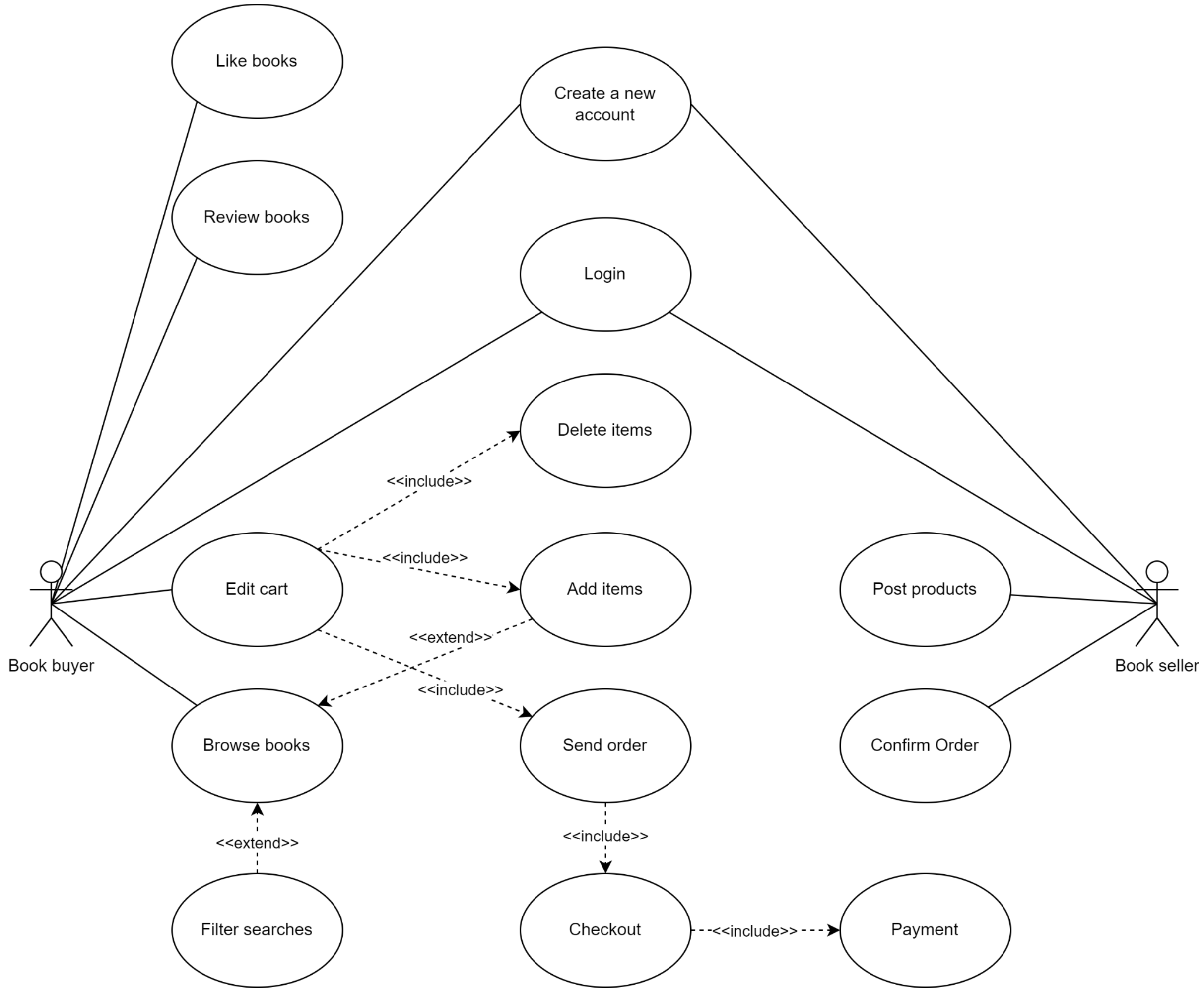
2.1 Requirements and constraints

- The user interface shall be implemented with responsiveness in mind, which ensures optimal performance on most common screen sizes.
- The device used to access the platform must have a screen, an input device, Internet connection and a supported web browser installed.
- The system shall be developed using React.js for the front-end and JSON server to simulate a back-end.

2.2 Quality ranges

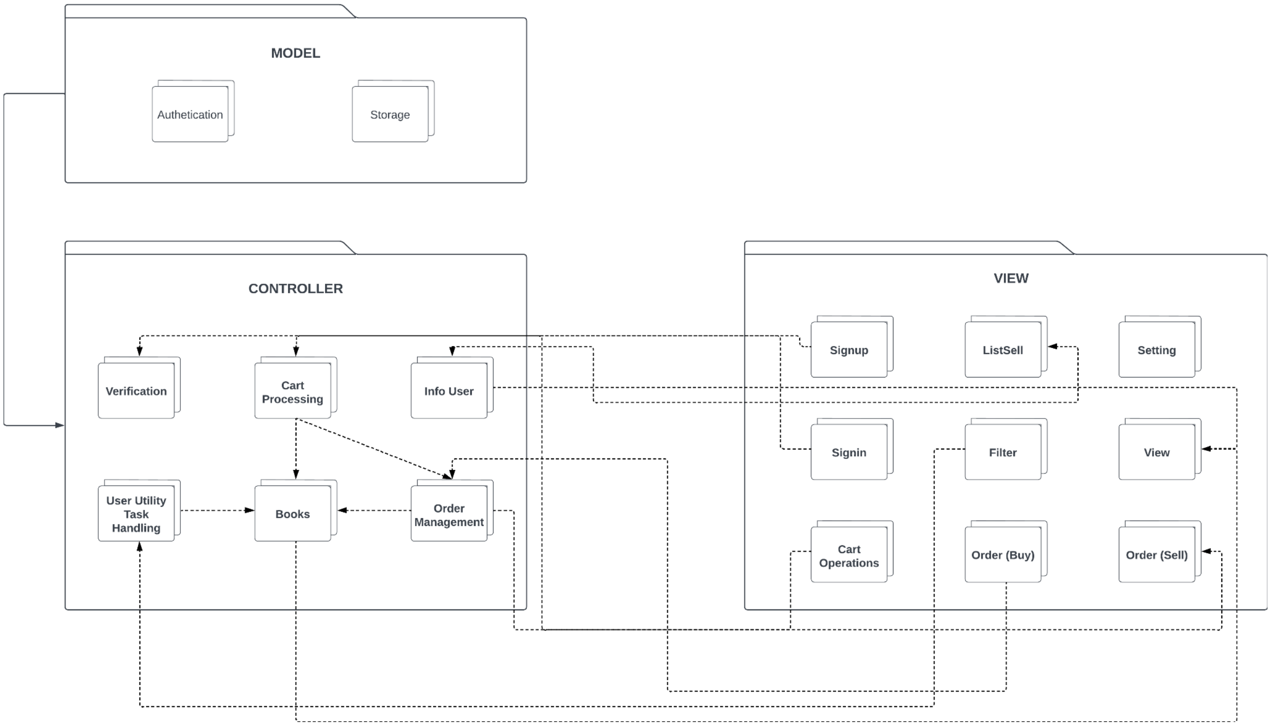
- Availability: The platform shall be available 24 hours a day, 7 days a week.
- Usability: The platform's user interface should have a level of usability appropriate for the literate and computer-literate population. Users should not require the use of a hardcopy manual to operate the platform.

3. Use-Case Model



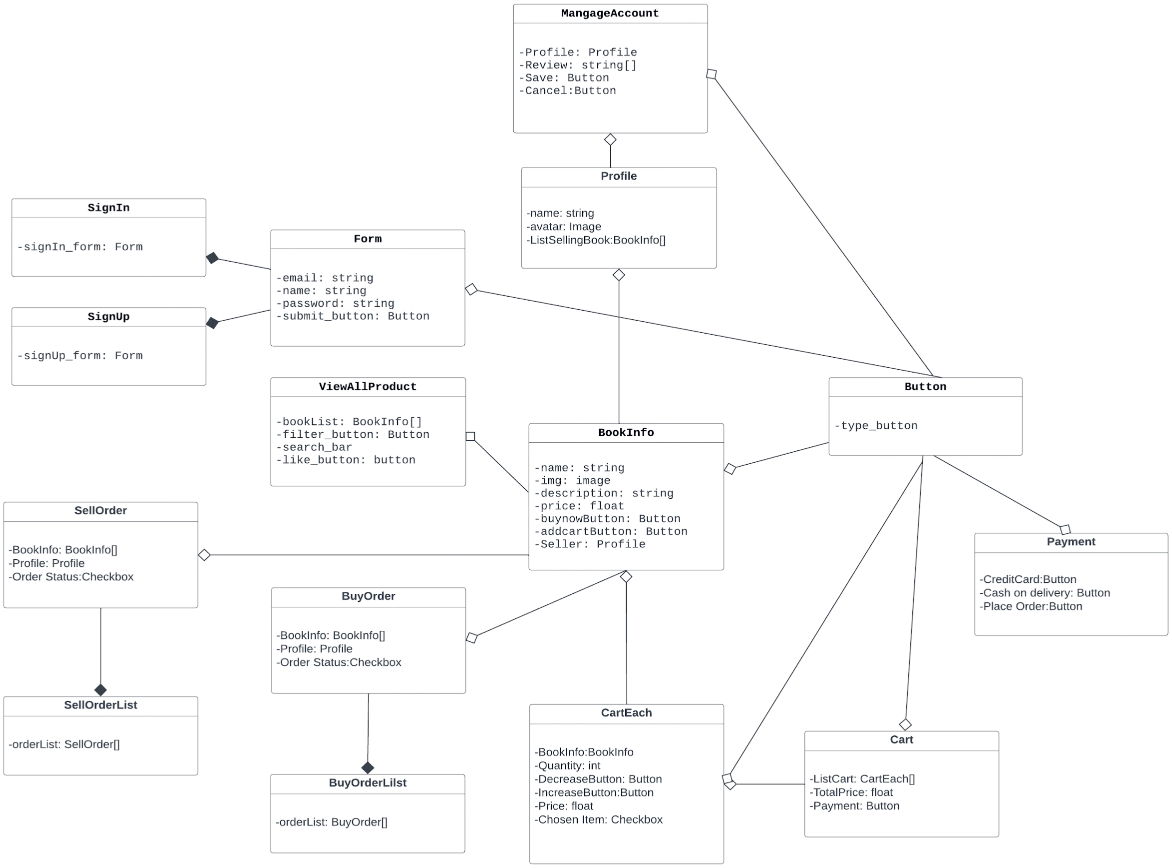
4. Logical View

Read & Relay's architecture is based on the MVC model, with three components: the user interface (view), the system logic (controller), and the database (model).



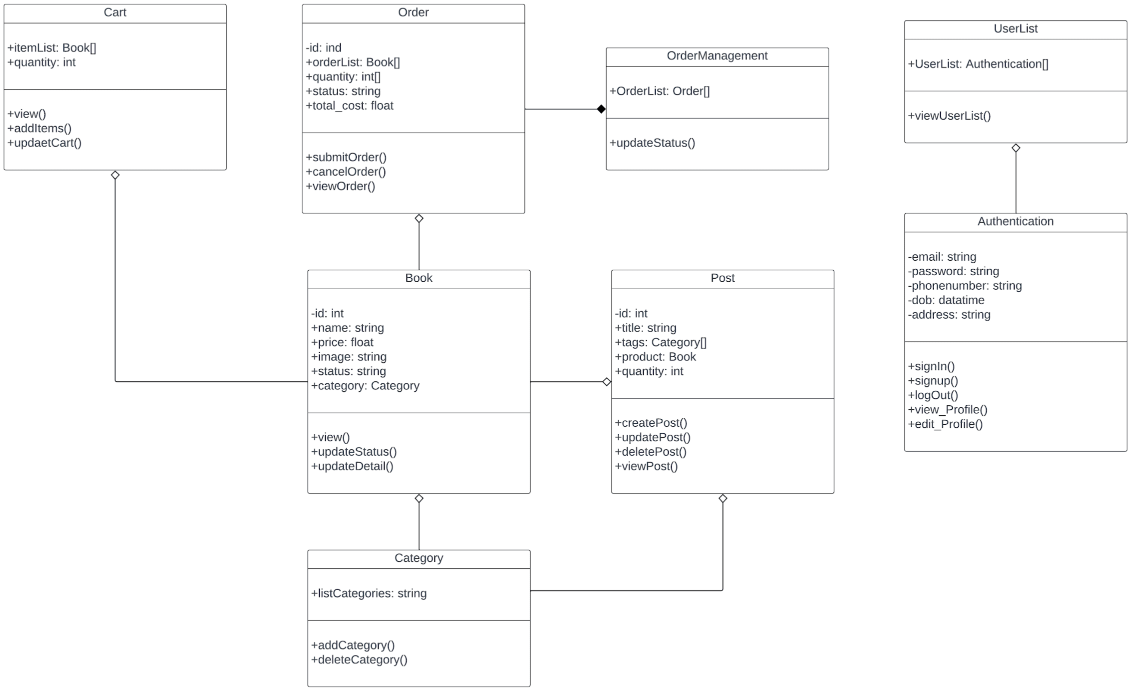
4.1 Component: User Interface (View)

The user interface is the only component visible to the user. By interacting with the elements on the interface, the user will send requests to the controller. The UI will be constructed using React.js.



4.2 Component: System logic (controller)

The controller will be handling and processing all the requests received from user interaction with the view component, either by supplying the corresponding screens/UI components, retrieving the necessary data from the model, or modifying the existing UI components on the interface. The back-end server will be constructed using Express.js. System logic will be developed using node.js.

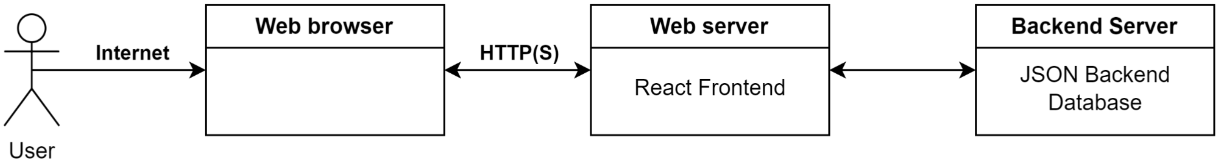


4.3 Component: Database

The database stores all data on users, books and orders. For demonstration purposes, the data used in Read & Relay is stored in a simple JSON file. JSON Server is used to simulate a RESTful API, and API calls are done via Axios.

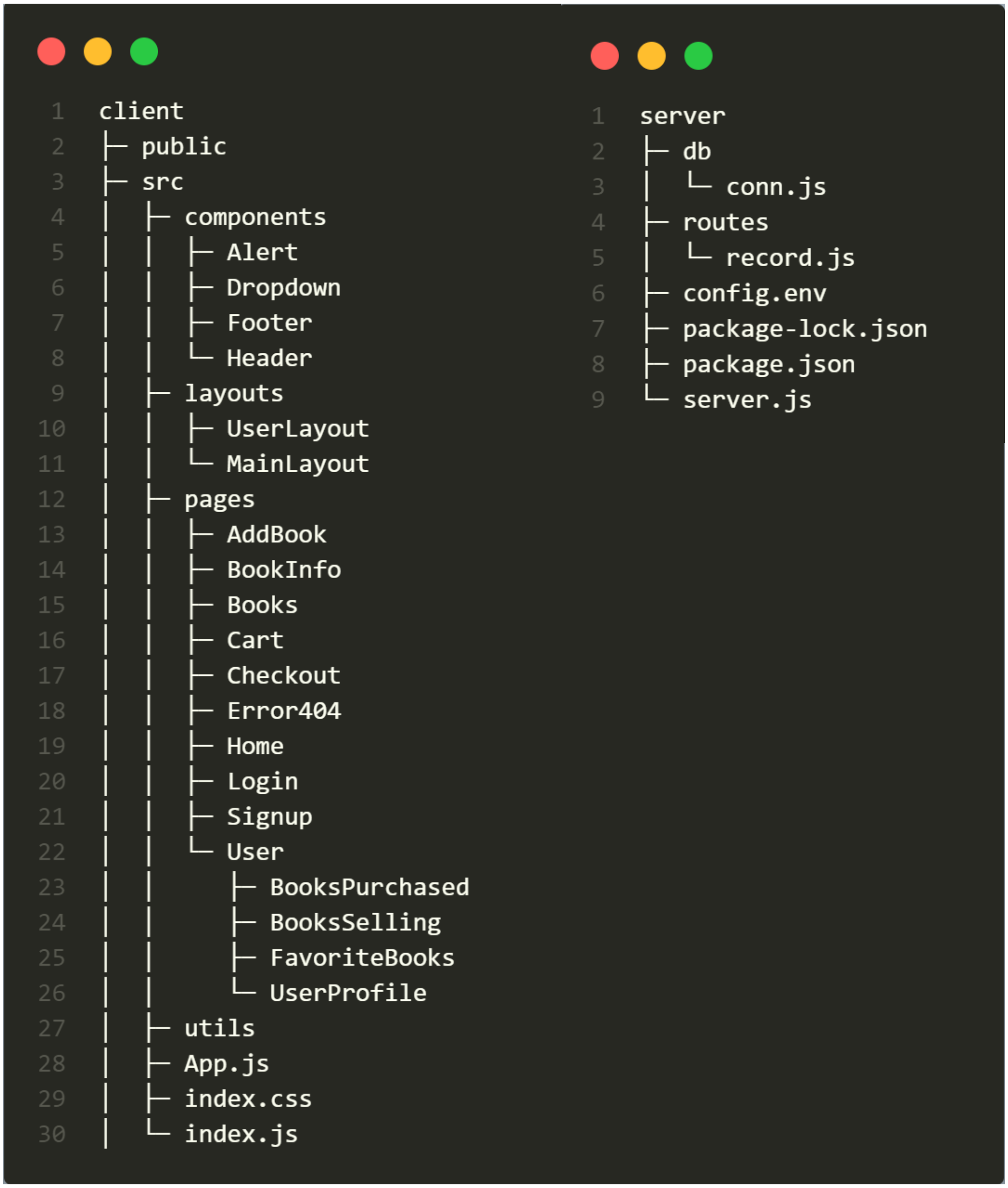
5. Deployment

Read & Relay online book-selling platform, built with React for the front-end and JSON for the back-end, allows users to access its services through web browsers on their personal devices via an Internet connection. The communication between the front-end and back-end involves the exchange of data in JSON format. This format is used to transmit structured information, such as book details or user profiles.



6. Implementation View

Below is the folder structure of the platform’s source code. The source code is divided into two main folders: the client folder and the server folder. The client folder consists of subfolders, defining the front-end components, layouts and pages, as well as other utility functions. The server folder consists of the “database”, stored in a JSON file, as well as other JavaScript files containing data processing functions.



```
client
├─ public
├─ src
│  ├─ components
│  │  ├─ Alert
│  │  ├─ Dropdown
│  │  ├─ Footer
│  │  └─ Header
│  ├─ layouts
│  │  ├─ UserLayout
│  │  └─ MainLayout
│  ├─ pages
│  │  ├─ AddBook
│  │  ├─ BookInfo
│  │  ├─ Books
│  │  ├─ Cart
│  │  ├─ Checkout
│  │  ├─ Error404
│  │  ├─ Home
│  │  ├─ Login
│  │  ├─ Signup
│  │  └─ User
│  │     ├─ BooksPurchased
│  │     ├─ BooksSelling
│  │     ├─ FavoriteBooks
│  │     └─ UserProfile
│  ├─ utils
│  ├─ App.js
│  ├─ index.css
│  └─ index.js
```

```
server
├─ db
│  └─ conn.js
├─ routes
│  └─ record.js
├─ config.env
├─ package-lock.json
├─ package.json
└─ server.js
```