HofTAPS

Online Textbook Exchange

Architecture and Design Specification

Awab Abedin, Robbie Bernstein, Thomas Catania, Sanjit Menon, James Nastasi,

Marcus Wiesenhart

Version 1 - 4/2/25

Table of Contents

1. Introduction

1.1 Purpose

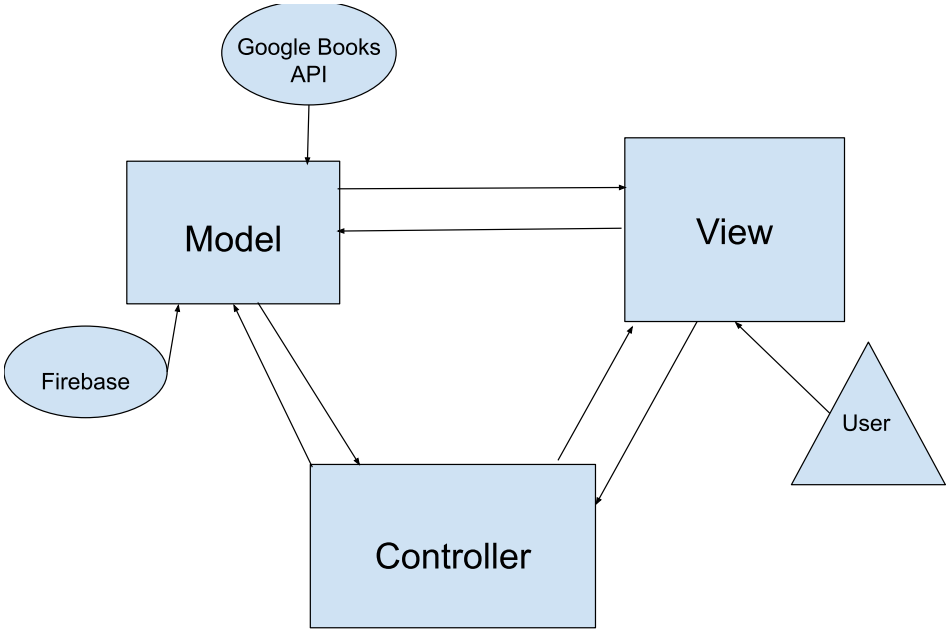
The purpose of the paper is to provide all architecture and design specifications that are used to develop the HofTAPS Textbook Exchange Platform. The app allows users to interact with each other in order to sell and purchase textbooks. This requires interactions between the users and the data from both Firebase and the Google Books API. The data that comes from the Google Book API is textbook image, title, and the author. The user inputs ISBN, price, condition, subject, and a brief description of the book. All of this information is then stored in Firebase. The application also supports user authentication and a notifications system that allows for users to see information on books they are selling and books that they have made an offer on.

1.2 Project Scope

The scope of this project is building a network of users that will interact with each other in a mini-marketplace for textbooks with the help of Google Books, which displays information about the textbook. The webpage allows verified users to list books, send an offer to buy a book, as well as create a wishlist of books they might be interested in. Interactions between users occur in a notification system that is built into the webpage. They are able to see if someone wants to buy their book, if a seller has accepted the offer to buy, if a textbook that you made an offer on has been sold to someone else, and if someone declined your offer.

2. System Overview

2.1 High Level Overview



The main architectural pattern implemented by HofTAPS is the Model View Controller design. This model allows for direct and clear communication between HofTAPS' user interface, database, and backend. The Model is made up of our Firebase database and the Google Books API. These are the main sources and storage for the data we are using. Firebase is responsible for storing all our user information as well as all textbook information entered by users. In addition, it handles user verification, authentication, and encryption, a safe and secure user experience. The Google Books API is responsible for gathering additional information on textbooks. The ISBN number entered by is passed to the API call, which returns all information about the book. We gather the title, author, and cover thumbnail to ensure accurate information for our users. The View includes all user interfaces on our platform. These are implemented using JavaScript, HTML, and CSS to provide a well-designed and easy-to-use interface for users. The Controller aspect involves all communication between user interactions with the database. These include adding textbooks to Firebase when a user posts a book, removing them when a user purchases, etc. These functions are implemented in JavaScript alongside most UI implementations. This made for easy integration between front-end and back-end functions.

2.2 Technical Characteristics

2.2.1 UI and User Functions: The HofTAPS UI allows users to sell and purchase textbooks easily. All front-facing design and interactions is implemented using JavaScript, HTML, and CSS with Bootstrap.

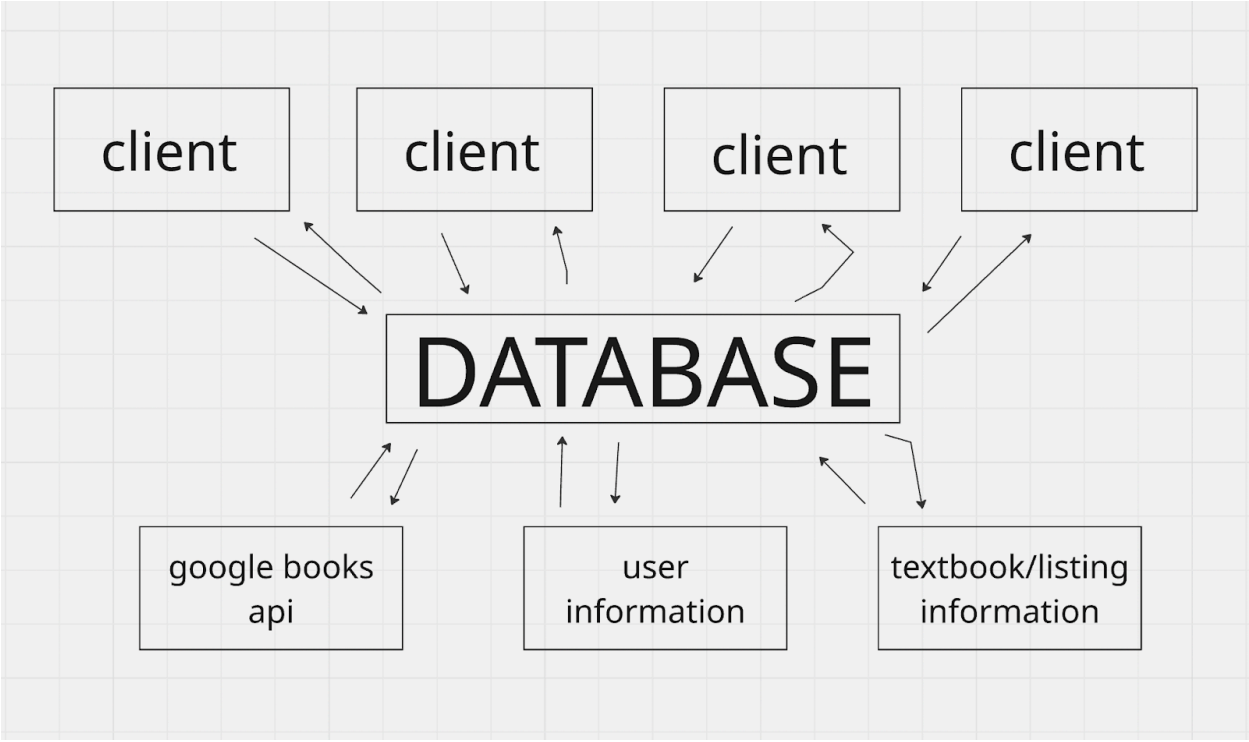
2.2.2 Database: Firebase's Firestore feature allows all user information and textbook information to be stored for easy access. Each category is stored in separate documents, each of which can be accessed individually.

2.2.3 API: Google Books API provides all the information for the textbooks. Using the textbook's ISBN number provided by the user, a call is made to the API which returns all relevant information about that book. Then, that information is stored in Firebase where it can be easily accessed.

2.3 Architecture Styles

The main architecture style, as described in section 2.1 above, is the MVC model. However, other styles are implemented as well. The styles include the Layered, Repository, Pipe and Filter, and Client Server models. All of these styles have their specific uses, which make them useful to be implemented in different areas for different functions. Each of these are described in further detail in Section 4.

3. Key Architectural Components



A very basic example of the Client-Server architecture that HofTAPS implements. Clients on the front end request and get data from the database, while the database (firebase) stores, manipulates and processes data. (does not include every aspect of the architecture, it is just a visual representation of the architecture).

3.1 Programming Environment

The HofTAPS team is using VSCode as the integrated development environment to develop the textbook exchange web application. VSCode provides free extensions that help the team members write, debug and test code. This IDE is also very accessible and intuitive so any level programmer can utilize it. In addition, GitHub is a great repository for sharing the team member's code and performing integration testing through cloning the repository. We used the main branch while also preserving older versions of the application locally on our computers (or on other branches). The website is primarily created with JavaScript/JSON handling the backend logic and design. We also are using functions from Firestore to help develop the application. On the front-end, HTML/CSS with Bootstrap as the framework helps make

the User Interface and visuals perform seamlessly for the user. JavaScript is also used on the front-end side of the application (e.g. an alert signaling that a book listing has been added to the user's wishlist).

3.2 Database

The database that handles all the HofTAPS' user data, security, and real time updates is Firebase. Through implementing the "Real Time Database" features that Firebase offers, our team can easily access and manipulate user data, transactions and metrics relating to textbook listings. Integrating the Google Books API with Firebase allows for new listings to automatically sync with real textbooks without the seller having to fill out all of the book details in the 'sell page', ensuring accurate information is displayed on the site. While minor tasks like wishlist management are done through JSON parsing, user data and order history are handled all on Firebase through Firestore.

3.2.1 User Data: All of this information is stored in an organized document in Firestore. This includes data given at signup such as email, first name, last name, and h700 number. All of this data can be changed in the database through the user profile.

3.2.2 Listing and Textbook Data: All textbook data, including each book's ISBN, title, author, and any associated images are stored in a document in Firebase.

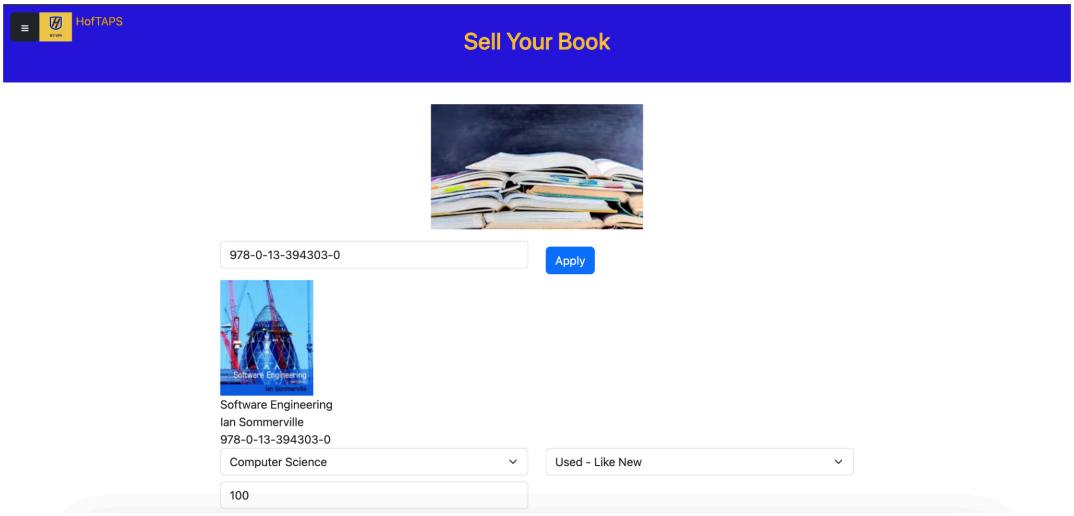
Certain limitations that arise when using a database like Firebase are data usage restraints and the overall reliance on the database to carry out and store all user data. Switching this project to another database, for example, would be a very daunting task. When it comes to data usage, if HofTAPS goes over a certain limit of data consumption, we would have to upgrade to a premium plan and invest money into keeping the application up and running.

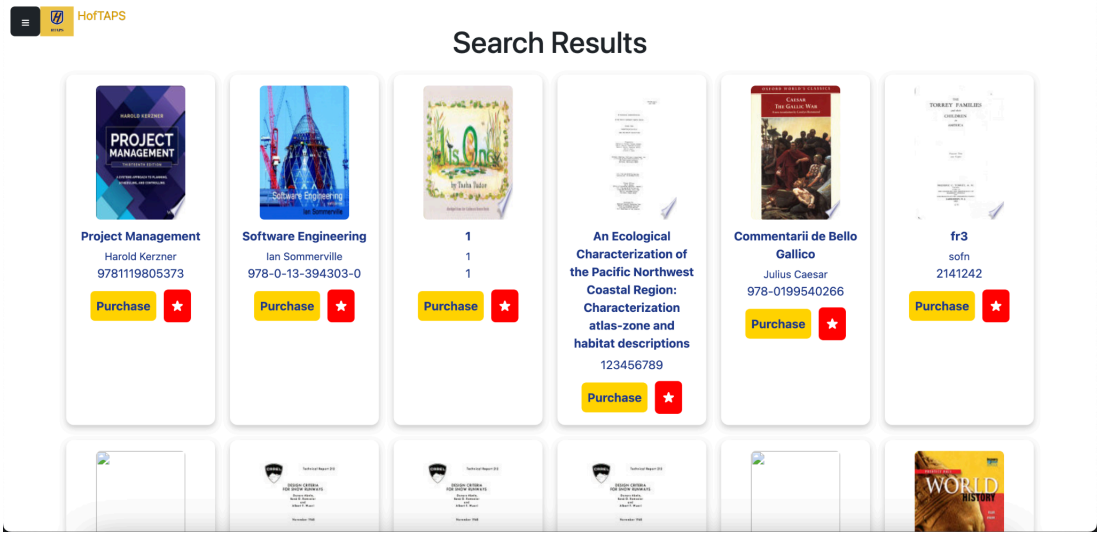
3.3 Email/Notification System

The notification system for HofTAPS is handled entirely within the application. Email notifications are not a design choice for HofTAPS with the only email received by the user being the verification link when they originally authenticate their account. This system means that order updates, offers and order confirmations will only be available for the user inside the notification section of the application. This design choice allows for a much easier path towards successfully creating a functional notification

system, however a challenge lies in ensuring that the user is signaled when they receive a notifcation. This design choice also puts heavy emphasis on making sure the web application functions as intended or else all notifications will be unavailable for the user to view.

3.4 Textbook/Listing Details





3.4.1 UI and User Functions: The Textbook Listing component provides an intuitive interface for users to create, view, and interact with textbook listings. Users can upload their textbooks with detailed information from Google Books API and browse available listings. The listing detail page displays comprehensive information about books, including the author,

edition, and book condition. Sellers can manage their active listings while buyers can add items to their wishlist or purchase a textbook.

3.4.2 Database: Textbook listings are stored in Firebase Firestore as individual documents within a "textbooks" collection. Each listing document contains fields for book information (title, author, ISBN), condition details, pricing, and seller information. The database structure allows for efficient querying by various parameters such as subject, price range, and condition. References to seller profiles are maintained to connect listings with user accounts.

3.5 Transaction Processing and Completed Exchange

The Transaction Processing component facilitates the entire exchange process between buyers and sellers without handling monetary transactions. When a buyer is interested in a textbook, they click the “purchase” button under the listing. The seller then receives an in-site notification letting them know someone wants to purchase their listing. They can then either accept, which completes the transaction, or decline the offer. All user interactions related to transactions occur exclusively within the application using our custom notification and messaging system. The app sends real-time notifications when offers are made, accepted, or rejected. No external communication platforms or payment processing systems are integrated. This keeps all transaction history contained within the application for user convenience and provides a complete record of exchanges

4. Architecture and Design Patterns in HofTAPS

Several Architectural and Design Patterns are utilized in our application. The MVC architecture was already discussed above in Section 2.1, however there are several others that play an important role in ensuring a well-organized structure and clear communication between all aspects of the application.

4.1 Repository Pattern

In the HofTAPS application, the Repository pattern is utilized for managing database interactions, ensuring that domain logic is clearly separated from direct database queries. This pattern is implemented through repository classes which contain all database operations related to textbook listings, user data management, and transaction records.

4.1.2 GitHub:The use of GitHub as a version control repository further enhances this pattern by logically organizing files and codebases into separate directories. Each repository on GitHub corresponds to specific application modules, such as user authentication (app.js), textbook management (googleBooksConfig.js), and notification systems (notification.js), allowing more organization collaboration. By utilizing branches, the team can isolate new features, bug fixes, and application prototypes without disrupting stable production code.

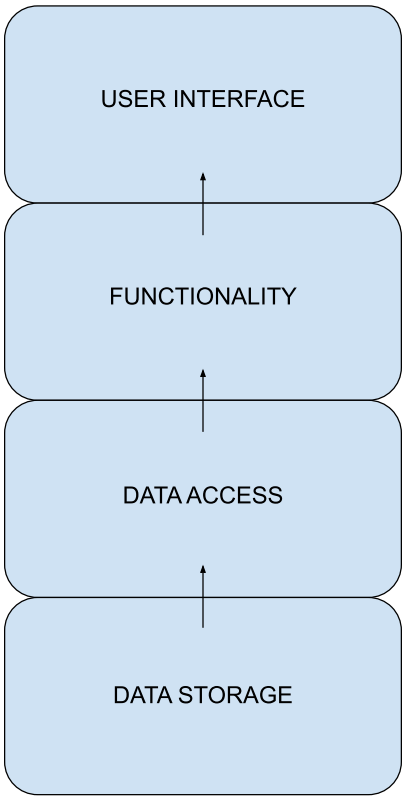
4.2. Client-Server Architecture

HofTAPS employs a robust Client-Server architecture, separating the application into frontend and backend components. The client-side, constructed with Bootstrap, HTML, CSS, and JavaScript, provides the user-facing interface allowing user interactions with the application. The backend, developed using Firebase, manages and processes user requests, authenticates users, and integrates external APIs like Google Books API to allow the application to make requests to the API. This architecture enhances scalability, allows for an increase in management of resources, and provides the flexibility to update client or server functionalities without impacting the entire system itself.

4.3 Observer Pattern

The observer pattern is utilized in HofTAPS through its notifications system. This design pattern enables dynamic notifications to users based on specific triggers such as textbook availability, wishlist updates, and interactions between the buyer and seller. Observers, like email (through Firebase) or in-app notification modules, are immediately informed whenever an event occurs within the system.

4.4 Layered Architecture



The HofTAPS application comprises several layers in terms of architecture to distribute responsibilities to each level of the system.

4.4.1 User-Interface Layer: The presentation layer is responsible for user interfaces and interactions. It manages the display of information, user navigation, and user input. Developed with Bootstrap, HTML, CSS, and JavaScript, this layer ensures that users are able to utilize an accessible interface.

4.4.2 Functionality Layer: The domain logic layer processes the core application rules and

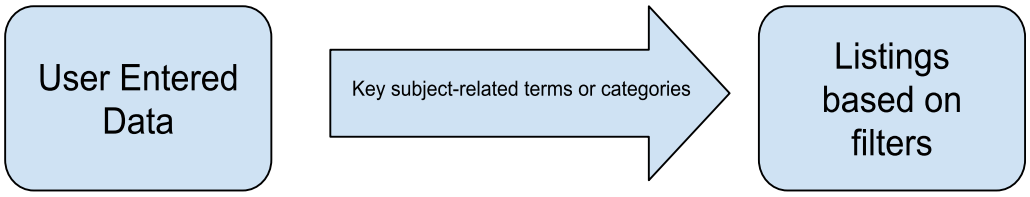
operations of HofTAPS. It includes functionalities such as user authentication, account management, and textbook listing operations. This layer consists of the rules which define the functionality of the application itself.

4.4.3 Data Access Layer: The data access layer utilizes operations to manage the database. For example, when a textbook is sold or marked as unavailable, the data access layer is responsible for determining operations to be performed in order to purge the information contained in the database itself.

4.4.4 Data Storage Layer: The data storage layer is responsible for securely storing user and textbook information. HofTAPS relies on Firebase, allowing the system to save user, textbook, and listing information in organized subcategories. Firebase allows for the simplification of data storage and retrieval, ensuring reliability and consistency.

This modular and structured design ensures changes in one layer minimally impact other layers, significantly enhancing maintainability and facilitating easier debugging and development.

4.5 Pipe and Filter Pattern



The Pipe and Filter pattern is utilized by HofTAPS through the copious amounts of search filters that the application implements. When a user initiates a search, the query passes through multiple filters including keyword matching and ISBN identification to output a list of textbooks. The system also implements several other filters to output a list of textbooks:

4.5.2 Search by Subject: When users perform a search by subject, HofTAPS applies the Pipe and Filter architecture in the following manner:

- Input Pipe:

        - User enters a subject-related query (e.g., "Computer Science" or "Mathematics").
    - Filters:
        - Subject Identification Filter:
            - Parses the query, identifies key subject-related terms or categories through the Firebase database.
        - Database Query Filter:
            - Passes the standardized subject to the database, retrieving textbook records matching the specified subject.
    - Output Pipe:
        - Presents filtered textbooks based on subject, sorted by relevance for accessibility

4.5.3 Search by Price: When users perform a search by subject, the Pipe and Filter architecture implemented in the following manner:

- Input Pipe:
    - User enters a price range or selects a price filter.
- Filters:
    - Price Parsing Filter:
        - Parses and validates the user's desired price range input, ensuring the values associated with “price” are correctly formatted.
    - Price Comparison Filter:
        - Queries textbook listings stored in Firebase to retrieve the listed textbooks that have the “price” attribute within the specified price range that the user requests.
    - Sorting Filter:
        - Arranges textbooks from lowest to highest asking price as specified by the user.

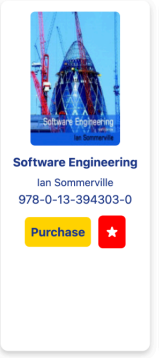
- Output Pipe:
    - Displays textbooks sorted by the given price criteria, enhancing convenience for the user.

4.5.4 Search by Condition: When users perform a search by condition, the specified architecture is implemented in the following manner:

- Input Pipe:
    - User selects a preferred textbook condition (e.g., "New," "Good," "Acceptable").
- Filters:
    - Condition Parsing Filter:
        - Validates the selected condition depending on what the user requests
    - Condition Matching Filter:
        - Searches Firebase database listings for textbooks that match the selected “condition” filter.
- Output Pipe:
    - Presents condition-specific textbook listings to users, allowing them to assess product quality for purchase.

5. Functionality Designs

5.1 Product Design



A product listing displays all relevant information about the textbook, including the title, author, and ISBN number. Each listing also features a thumbnail image of the cover of the book. The price of the listing is also displayed. There are two buttons associated with each listing: the first is the purchase button, which allows the buyer to request to purchase the textbook, and a wishlist button, which adds the textbook to the user's wishlist.

5.2 Textbook Offer Design

When a buyer wishes to purchase a textbook, clicking the purchase button initiates the sale. This sends a notification to the seller notifying them that someone wishes to purchase the textbook. The seller can then decide to accept or reject the purchase. The buyer will then receive a notification of the seller's decision. If the seller accepts, the transaction is completed and the textbook is removed from the database.

5.3 Product Categories

All textbooks feature certain information. Every user is required to enter the ISBN number, which is unique for every textbook. This ISBN is used to fetch the title and author from the Google Books API. The seller then enters the price they wish to sell the book for, along with the subject category, condition of

the book, and a short description. They can also upload images of their textbook so potential buyers can see the textbook and evaluate its condition.

Users searching for textbooks can filter and sort their searches by certain criteria. Filters include by min or max price and by subject, while they can also sort the searches by price.