HofTAPS

Online Textbook Exchange

Architecture and Design Specification

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

Marcus Wiesenhart

Version 2 - 4/22/25

Table of Contents

Cover Page...1

Table of Contents...2

1. Introduction...4

1.1 Purpose...4

1.2 Project Scope...4

2. System Overview...5

2.1 High Level Overview...5

2.2 Technical Characteristics...6

2.2.1 UI and User Functions...6

2.2.2 Database...6

2.2.3 API...6

2.3 Architecture Styles...6

3. Key Architectural Components...7

3.1 Programming Environment...7

3.2 Database...7

3.2.1 User Data...8

3.2.2 Listing Data...8

3.3 Notification System...8

3.4 Listing Details...9

3.4.1 UI and User Functions...10

3.4.2 Database...10

3.5 Transaction Processing and Completed Exchange...11

4. Architecture and Design Patterns...11

4.1 Model-View-Controller...12

4.2 Client-Server...13

4.3 Repository...14

4.3.1 GitHub...14

4.4 Observer…………………………………………………………………………………..………….15

4.5 Mediator……... ………………………………………………………………………..………...15

5. Functionality Designs...…………………………………………………………..………………………16

5.1 Product Design ………………………………………………………………………………….16

5.2 Textbook Offer Design……………………..……………………………………………………...17

5.3 Product Categories……………………………………………………………………………...17

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, and if someone declined your offer.

2. System Overview

2.1 High Level Overview

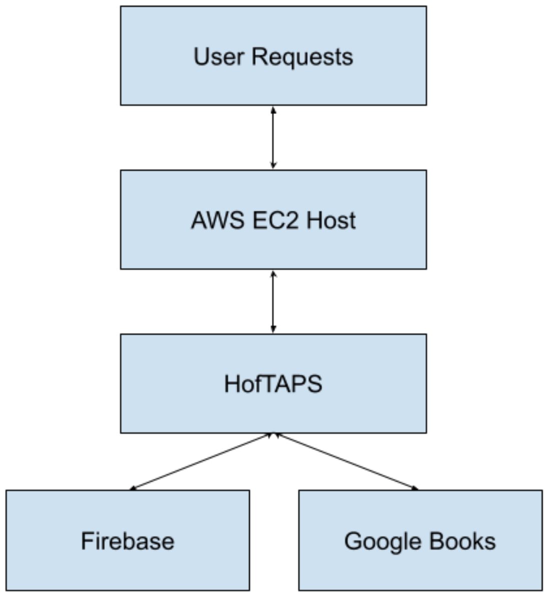


Figure 1

Pictured above is a visual representation of the high-level layout of the HofTAPS application. There are five main components: the HofTAPS source code, our Firebase database, Google Books as our API, an AWS EC2 instance for hosting, and the user. Users are able to view and access the HofTAPS site through an AWS server, allowing anyone to connect from anywhere. Our code base is what handles all user requests and calls to our database and API. Simple requests such as changing pages are all handled within our site, however certain requests require communication with Firebase or Google Books. HofTAPS takes the information the user enters and queries Firebase using that information. For example, search requests take the user input and requests from Firebase all textbook documents containing that input as a title, author, or ISBN. This information is then returned to HofTAPS, where it can be displayed for the user to browse. Interactions with Google Books happen in a similar way. A user enters a textbook ISBN, which is then sent in an API request to Google Books. Google Books will then return their matching entry to HofTAPS, where we parse it and display the information we need to the user. This information is then sent to Firebase where it can be stored and accessed later.

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, textbook, and notification 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

There are three main architecture styles utilized in HofTAPS. They include the Model-View-Controller, Client-Server, and Repository. The MVC model can be used to represent HofTAPS at a high-level. It controls how the user interacts with the site, how the site handles user requests, and how the site interacts with Firebase and Google Books. The Client-Server model shows the interactions between the users of HofTAPS, the AWS EC2 instance that hosts HofTAPS, our site, and Firebase and Google Books. The Repository model is employed in the form of GitHub, where all of the HofTAPS team members push and pull files to update or make changes as we build our system. These applications are discussed in further detail in Section 4.

3. Key Architectural Components

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 programmer can utilize it. In addition, GitHub is a great repository for sharing the team member's code and performing integration testing through repository cloning. We utilize the main branch while also preserving older versions of the application locally or on other branches. The website is primarily created with JavaScript/JSON handling the backend logic and design. We also utilize 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 to perform certain functions.

3.2 Database

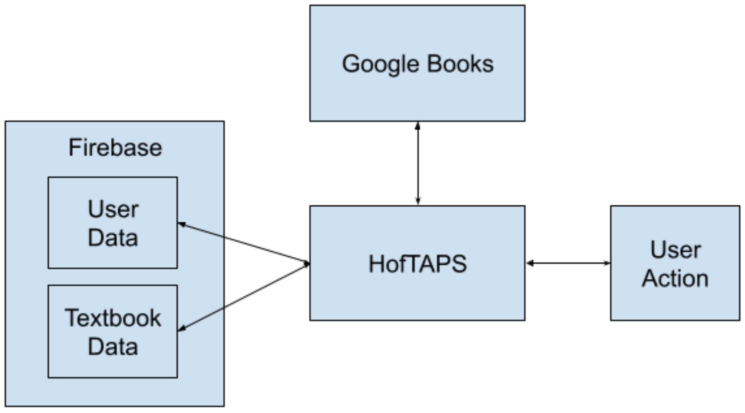


Figure 2

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. It is important to note that, as shown in the image above, Google Books and Firebase do not interact with each other directly. All calls to Google Books are made through HofTAPS, where they are transformed and sent to Firebase. 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 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 Notification System

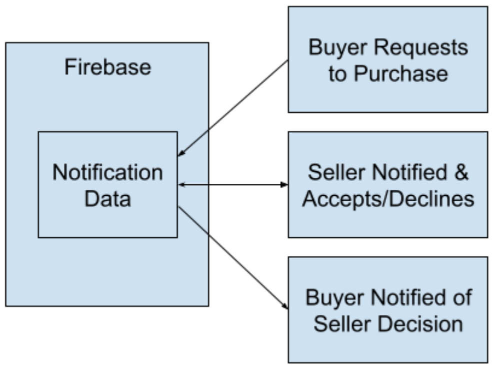
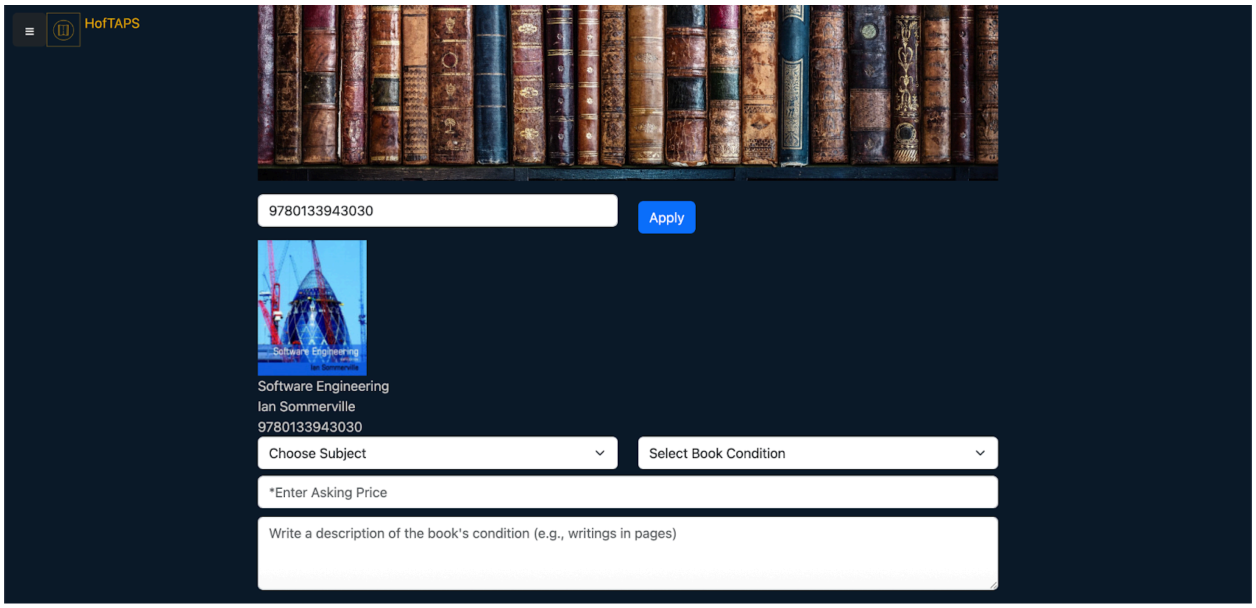

Firebase
Notification Data
Buyer Requests to Purchase
Seller Notified & Accepts/Declines
Buyer Notified of Seller Decision

Figure 3

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. Once an interested buyer requests to purchase a textbook, a notification document is created inside Firebase. This contains the buyer as the sender and the seller as the receiver. Each user's notifications inbox pulls out any notifications from Firebase that contain them as the recipient and are displayed. The seller then accepts or rejects the request, and another notification is created with the decision with the seller as the sender and the buyer as the recipient. 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.

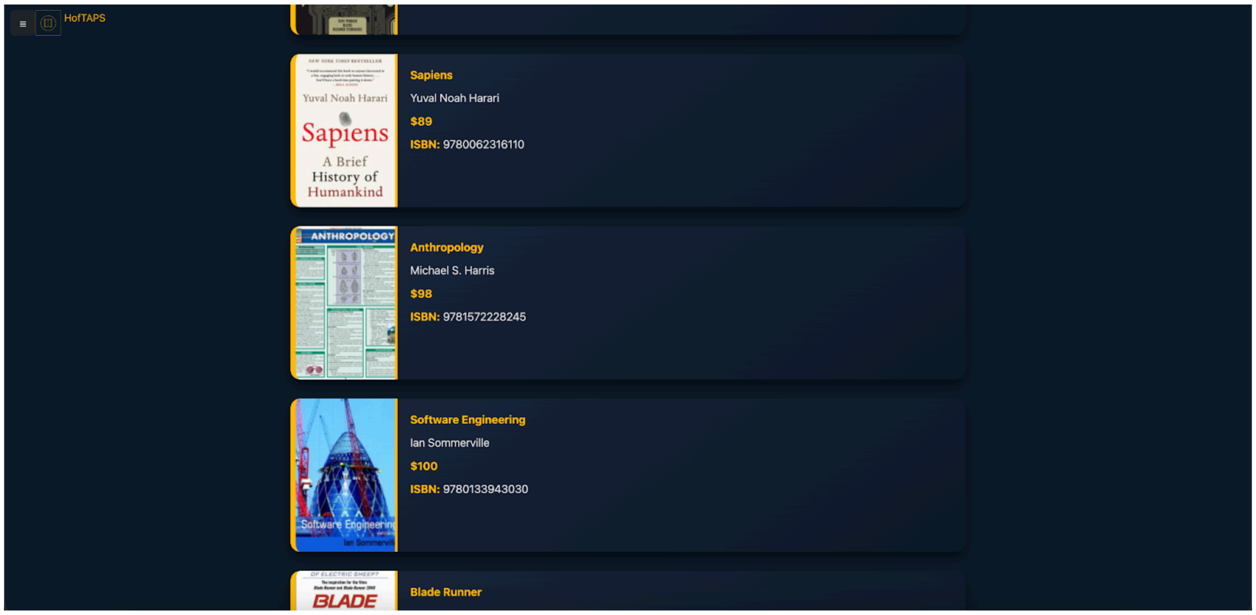
3.4 Listing Details


HofTAPS
Apply
Choose Subject
Select Book Condition
*Enter Asking Price
Write a description of the book's condition (e.g., writings in pages)

9780133943030

Software Engineering
Ian Sommerville
9780133943030

Figure 4



HofTAPS

Sapiens
Yuval Noah Harari
$89
ISBN: 9780062316110

Anthropology
Michael S. Harris
$98
ISBN: 9781572228245

Software Engineering
Ian Sommerville
$100
ISBN: 9780133943030

Blade Runner

Figure 5

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 (Figure 4). The listing detail page displays comprehensive information about books, including the author, edition, and book condition (Figure 5). 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

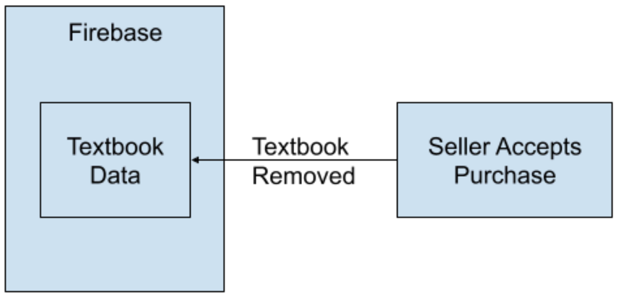


Figure 6

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 the request, which completes the transaction, or decline the request. 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. These interactions can be seen in Figure 3. 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. Once the seller accepts a purchase, the textbook is automatically removed from Firebase, as shown in Figure 6. The transaction is then complete.

4. Architecture and Design Patterns

Several Architectural and Design Patterns are utilized in our application. These patterns may not represent the application as a whole, but rather specific functions or structures.

4.1 Model-View-Controller

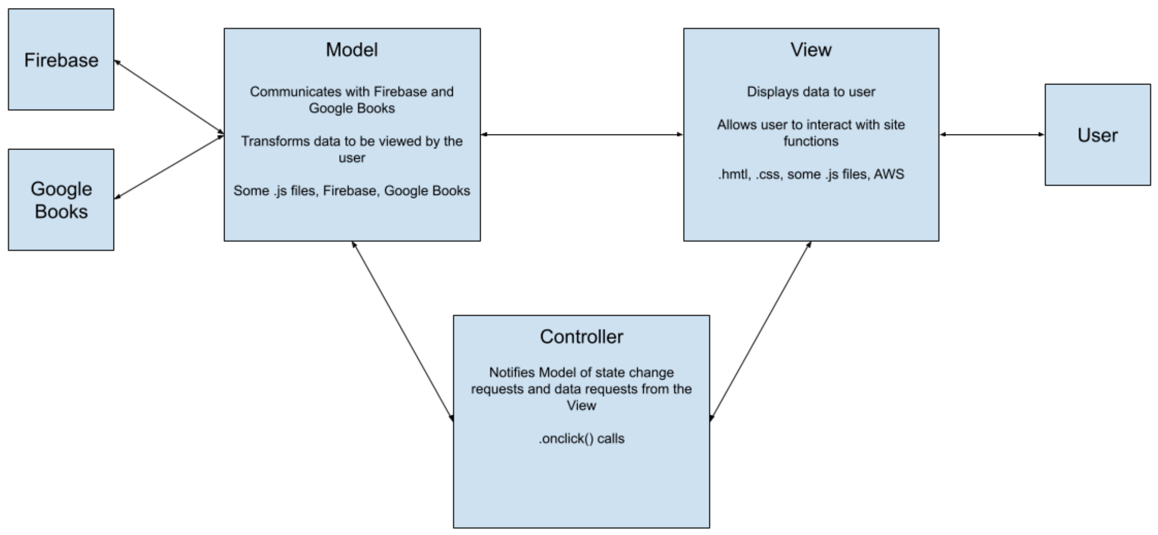

Firebase
Google Books
Model
Communicates with Firebase and Google Books
Transforms data to be viewed by the user
Some .js files, Firebase, Google Books
View
Displays data to user
Allows user to interact with site functions
.hmtl, .css, some .js files, AWS
User
Controller
Notifies Model of state change requests and data requests from the View
.onclick() calls

Figure 7

The MVC pattern can be used to describe the HofTAPS application as a whole. The model handles all interactions with Firebase and Google Books, including user authentication and textbook uploads and searches. The View displays information from Firebase to the user. Users see their account information and their current listings, and can search for textbooks and see their search results. The Controller handles user requests and state changes. These are often implemented using JavaScript's .onlick() method, which signifies that the user has clicked on some element. These control uploading or purchasing a textbook, site navigation, user sign-ins, and other user actions.

4.2. Client-Server

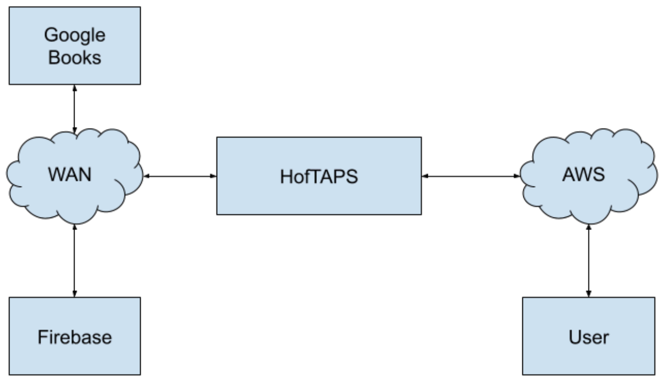


Figure 8

HofTAPS employs a robust Client-Server architecture, separating the application into frontend and backend components. There are two different client-server structures occurring at the same time. There is the interaction between HofTAPS and the user, with the user as the client and HofTAPS being hosted on AWS as the server. Users can view the site, navigate it, and perform certain functions. These functions are carried out on the server, and the results are then sent back to the client. There is also the interaction between HofTAPS and Firebase/Google Books, with HofTAPS being the client in this case and Firebase and Google Books as the servers. As the client, HofTAPS can request or send data to the server. All that is returned are results, with all computations taking place on the servers. Both of these models are “thin-client” models, with all computations occurring on the server and only display processing occurring on the client.

4.3 Repository

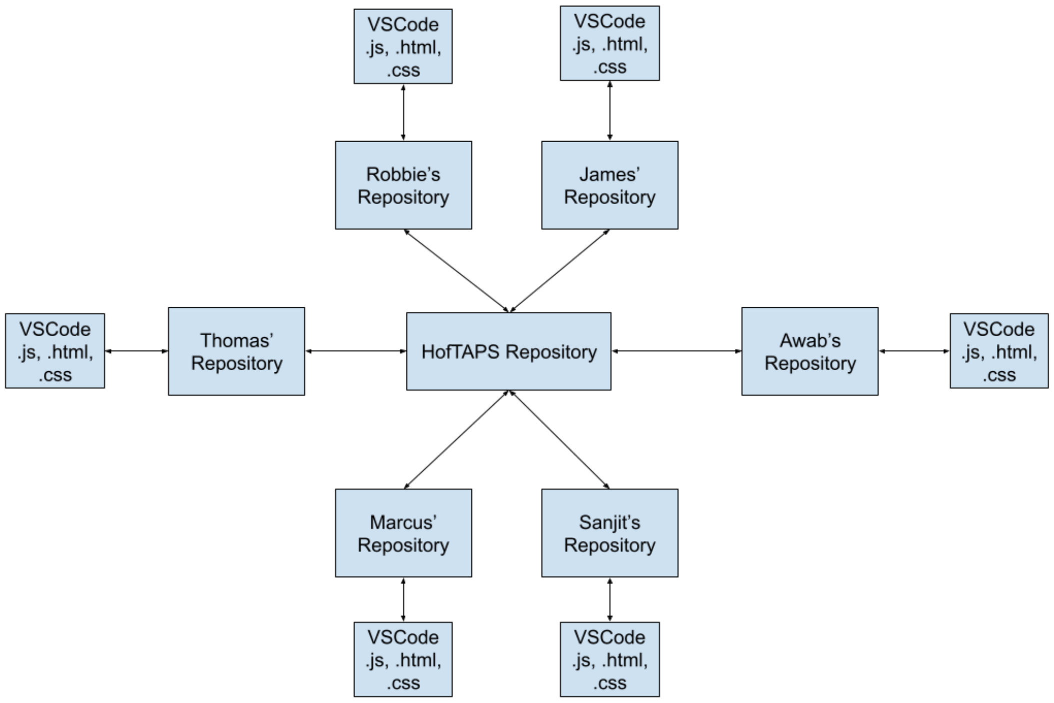


Figure 9

In the HofTAPS application, the Repository pattern is utilized for managing application development. It ensures each team member has access to the code base, can make changes and push those changes into the main source. It ensures proper compatibility between member's components and allows for easy software version management.

4.3.1 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.

4.4 Observer

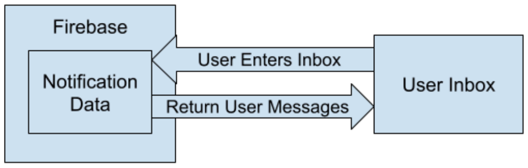


Figure 10

The observer pattern is utilized in HofTAPS through its notifications system, which is described in detail in Section 3.3. When a user enters their inbox, a query is sent to Firebase that pulls all notifications which list the current user as the recipient and displays them in the inbox. These notifications are available whenever they are requested by the user, with no delay between request and receival.

4.5 Mediator

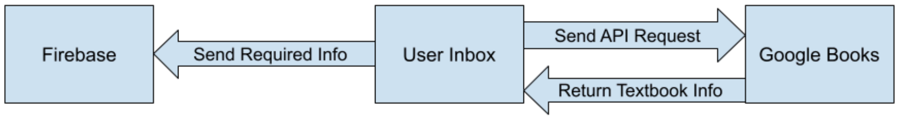
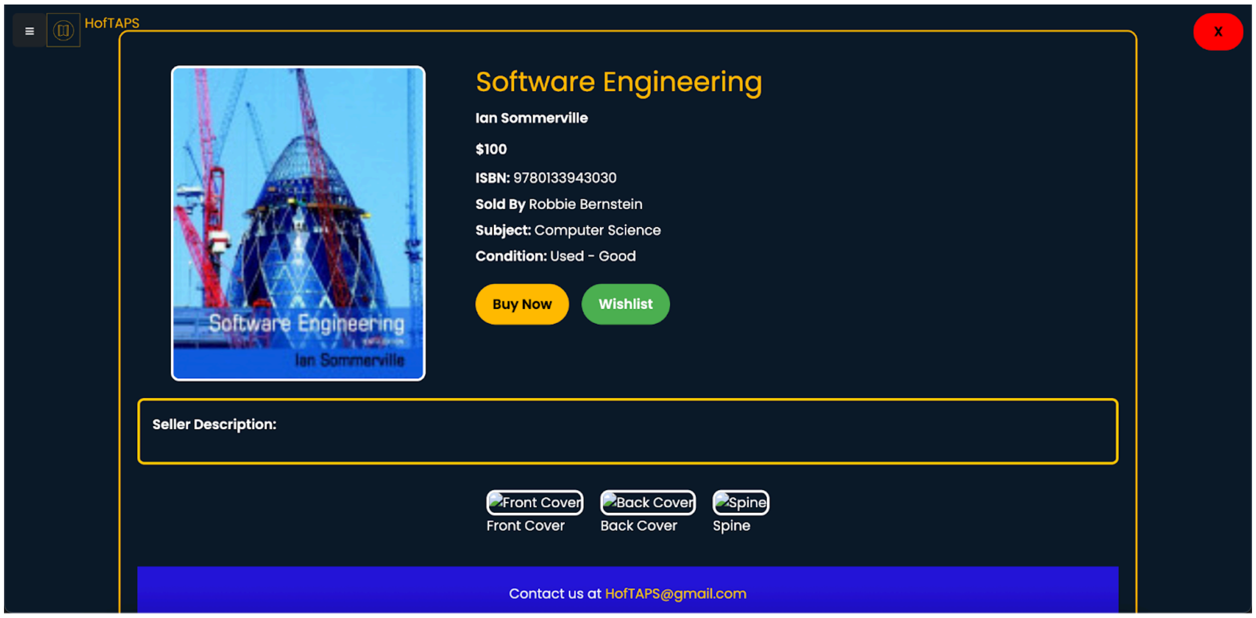


Figure 11

The mediator pattern is utilized in HofTAPS in the interactions between Google Books and Firebase. When a user enters an ISBN, an API call is sent to Google Books to retrieve the information for that book. Google Books returns extensive information about each book, however we only need the title, author, and the textbook thumbnail. These are extracted and are added to a new textbook document inside Firebase.

5. Functionality Designs

5.1 Product Design



Software Engineering

Ian Sommerville

$100

ISBN: 9780133943030

Sold By Robbie Bernstein

Subject: Computer Science

Condition: Used - Good

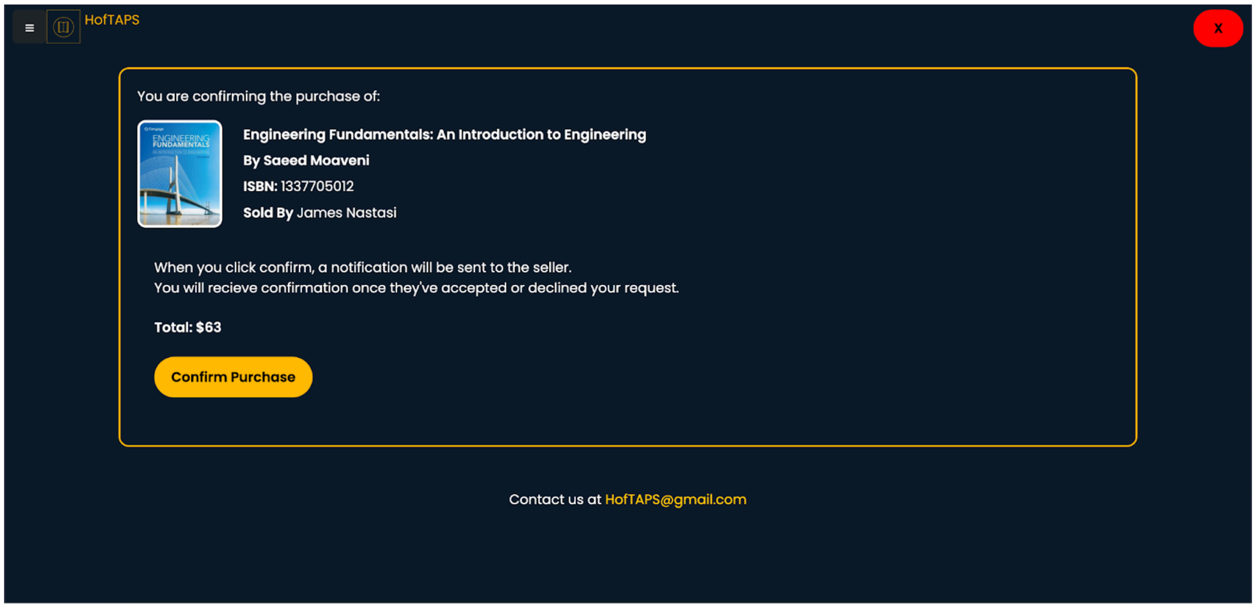
Seller Description:

Contact us at HofTAPS@gmail.com

Figure 12

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



You are confirming the purchase of:

Engineering Fundamentals: An Introduction to Engineering

By Saeed Moaveni

ISBN: 1337705012

Sold By James Nastasi

When you click confirm, a notification will be sent to the seller.
You will recieve confirmation once they've accepted or declined your request.

Total: $63

Contact us at HofTAPS@gmail.com

Figure 13

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.