IERG4999 SJ01 Final Year Project II

FYP Report with Final Consolidated Logbook

Cross-platform Application Development

Group: S1

1155144676 Yeung Tang

Supervised by: Professor Zhang, Kehuan

Date: 21/4/2023

A FINAL YEAR PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF BACHELOR OF INFORMATION ENGINEERING
DEPARTMENT OF INFORMATION ENGINEERING
THE CHINESE UNIVERSITY OF HONG KONG

Table of Contents

1. Abstract

Passing reading materials such as UGFN/H books from senior to junior years is a tradition in CUHK, but due to COVID-19, it is harder to meet new friends[1]. As a result, most students after finishing the course, simply throw away the textbooks as they have no one to pass them on to. This results in a waste of paper and money since those books are quite thick which consumes a lot of paper to print and it cost around HKD 100 per book.

To address the waste and encourage students at CUHK to meet each other, we have decided to develop an application that allows people to trade things they no longer need for something useful while promising social interactions and a chance to make new friends. Our application also promotes a green campus[2].

2. Introduction and Background

During the pandemic, students found it difficult to make new friends and maintain a normal social life due to restrictions on face-to-face interactions and the need to wear masks [1]. In a study conducted among universities, 86% of the students reported feeling socially isolated in the United States at the beginning of the pandemic [2]. Many students have then turned to social media applications to connect with friends and others. As a result, 70.8% of the students stated that the time they spent on social media increased during the COVID-19 pandemic [3].

As the trend of using social applications and to prevent waste on campus, we have decided to develop a cross-platform application that encourages students to exchange their 'waste', primarily in the form of books while chatting and meeting new friends.

Through the project, we have developed and deployed the Book Trade App on both Android and iOS platforms successfully, which achieves one of the goals for this project, creating a cross-platform Application.

The next goal achieved is that we have implemented the Chatter (Chat Room) function for users to chat and have social interactions with other users. We also

implement the TradeBucket (Trade Select List) for users to trade with other users conveniently for making counteroffers or setting up a deal.

Book recommendation from Chat GPT-3 is also implemented for users to seek a book that they are interested in.

Overall this project is a success and we have met our goals. We have developed and deployed a cross-platform book trading App, implemented social interaction features like chat rooms, Trade functions and UI's for users to trade books and prevent book waste, provided book recommendations to enhance the user experience and bring convenience to them. The project has achieved all the set objectives.

3. Methodologies

This project contains three main components. The first part is the UI/UX, which is the front end for users to interact with. The next part is the Node.js part, which provides APIs that help us create, read, update, and delete data. The final part is the MongoDB database to store user data and links for accessing the images.

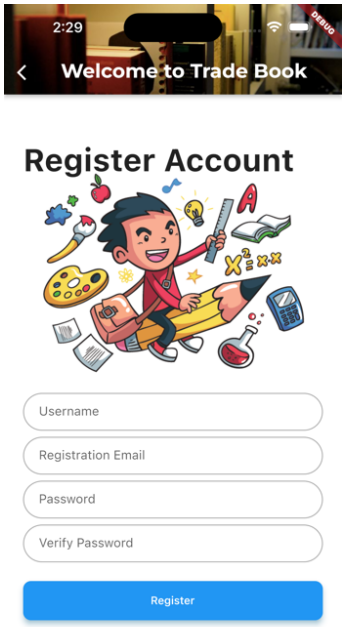
3.1 UI/UX for a user to interact with

The client-side application is mainly used by the end user. End-user may create their account and view their account details. Other than managing their account, the app serves as the medium to communicate with the server, e.g., users can submit photos and upload books, or real-time messaging with other users.

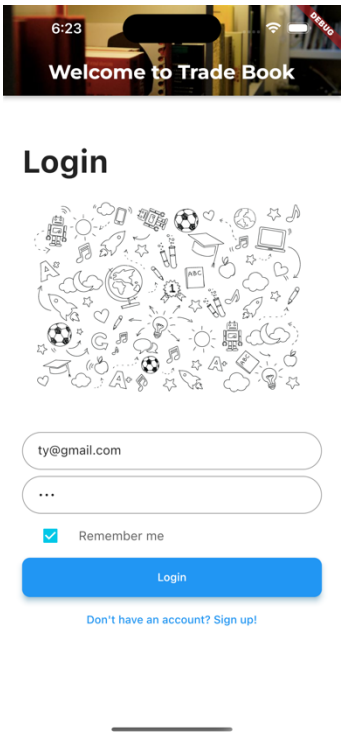
We developed the application in Visual Studio using the Dart programming language for the core logic. We also leveraged online UI design tools to create the user interface.

The application consists of multiple screens that serve different purposes and provide key functions, we will now walk through these screens and explain their functionality:

3.1.1 Register Page



This page is for new users to register their accounts. They simply need to enter their username, email, and password, and their account will be registered. If a user enters a duplicate username or email, the system will notify them that the account already exists and prompt them to choose a different username or email that hasn't been taken.

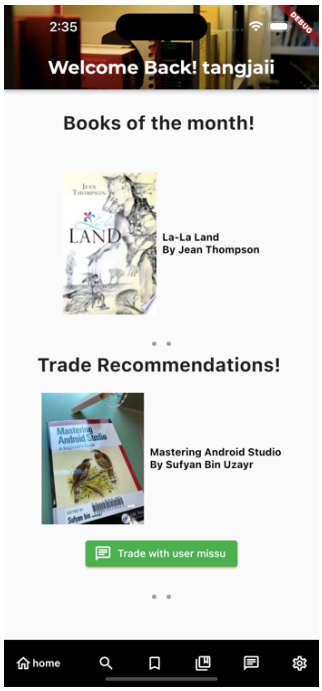


3.1.2 Login Page

This page is designed for users to log in to their accounts. To do so, users will need to enter their email addresses and passwords. Once the user inputs their credentials, the system will verify if the user has entered the correct information. If the credentials are correct, the user will be redirected to the home page. If the user selects the "Remember me" checkbox, their credentials will be saved on this device, and they will not need to re-enter their credentials again.

This page also includes a button for new users to register for a new account.

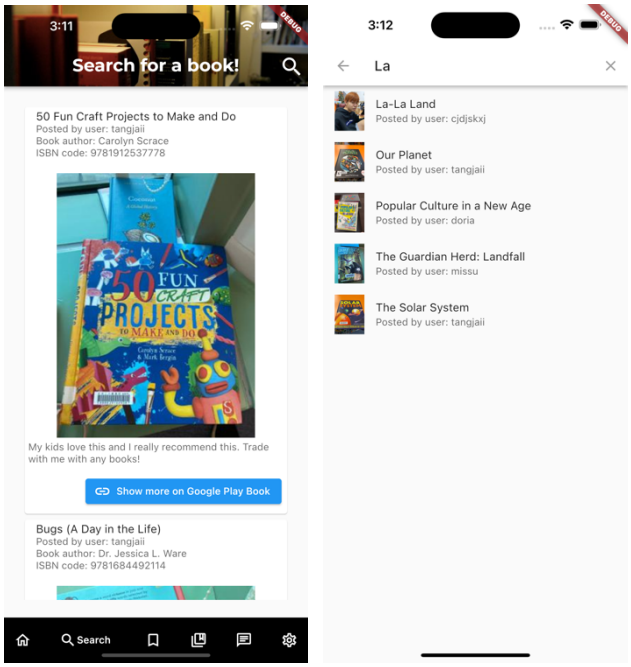
3.1.3 Home Page



The home page displays books that we recommend to users, as well as books that are currently popular. Through this page, users can easily find books that interest them. For both rows, a carousel slider is used to display a list of books.

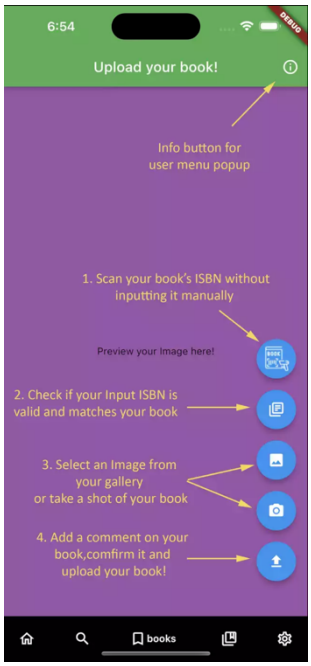
When the user clicks on a book, it will prompt up a browser and shows detailed information about the book on Google Book Store. If the user clicks on the "Trade with user xxx" button, it will then open the ‘tradebucket’, which is a part of the trade interface for the user to select books to trade with another user.

3.1.4 Search Page



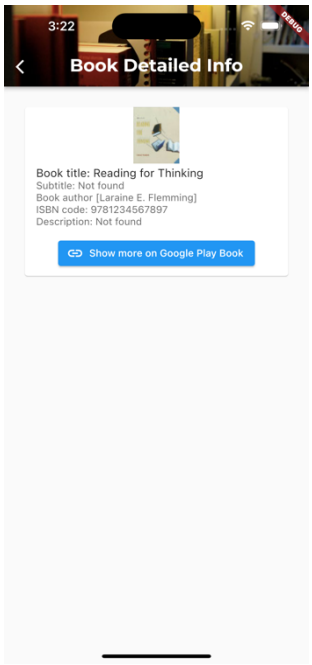
The search page is where users can search for books in our database. If any user has a book that is available for trade, they can search for the book here by clicking the search icon in the top right corner. Upon clicking the search icon, a page will slide up for searching the book name.

Users can also scroll through the search page without inputting any keywords to browse all of the available trade books.



3.1.5 Upload Page

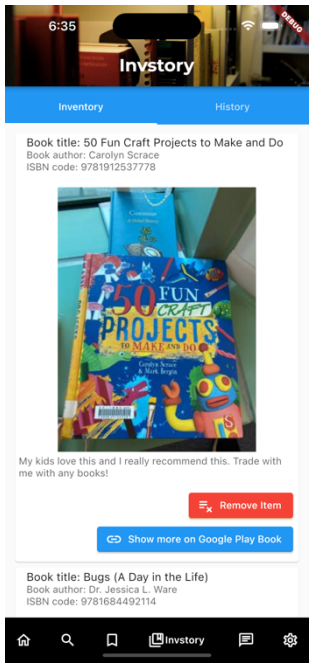
This page is mainly for users to upload their books to our database so that we can record them and make the books available for trade. It consists of 5 buttons: Scan ISBN, Scanned Book Details, Image Picker, Camera and Upload. If the user completes the process and presses the "Upload Book" button, the book with the image, ISBN, and user credentials will be uploaded to our database. Details of each button and APIs related will be shown during the final report.

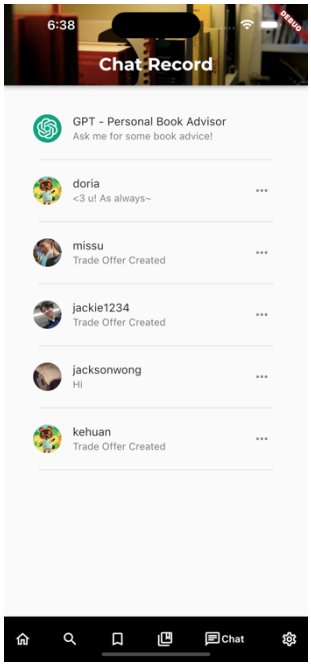


Scanned book detail example:

3.1.6 Invstory Page

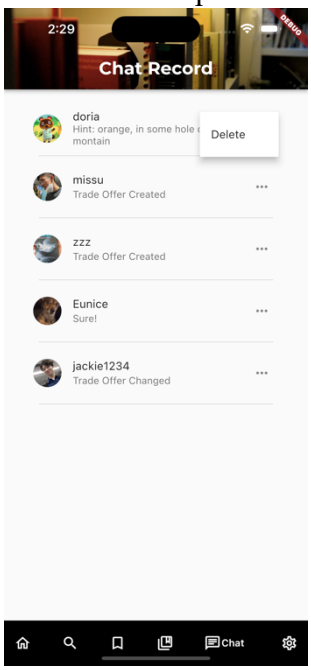
This page is intended for users to browse and modify their inventory, as well as the books that have been traded to other users. From this page, users can edit the comments and tags on their books, view more details about their uploaded books on Google Book Store, or remove their uploaded books from their trade inventory. The "History" tab allows users to browse the books that have already been traded to someone.





3.1.7 Chat Record Page

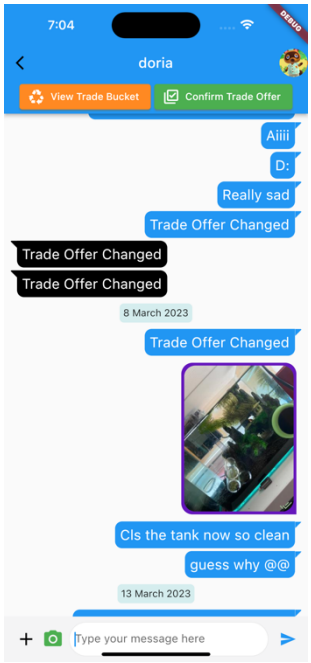
This page displays the chat contacts of other users if a chat record was found for the user. The user can also delete the chat record by clicking on the three dots at the end of each chat contact and selecting the "Delete" button. We also implemented a book recommendation chat feature for users to search for books that they want. Details about the "Read with GPT" page will be provided below.



How to delete a chat:

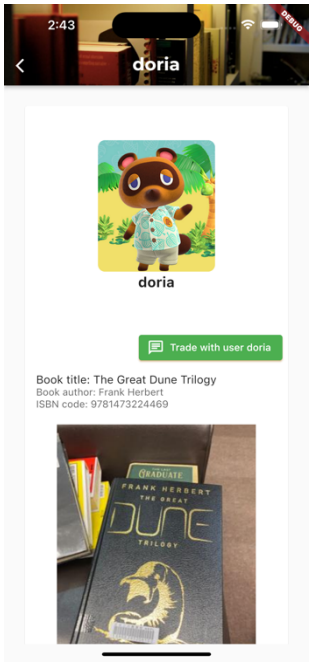
3.1.8 Read with GPT Page

This is a page that provides book recommendations based on the tags that the user enters. Once the user inputs the tags and presses the button, the keywords are sent to GPT, which then returns some results. After receiving the results, we will need to verify them since GPT is known to provide inaccurate results (Fake Data). Once the results are verified, they will be displayed in a slideshow with a maximum of three books, and the user can click on the slideshow for more details about the book. The validation process will be discussed in later chapters of this report.



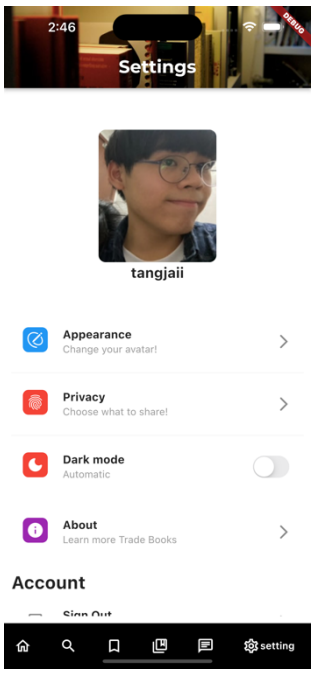
3.1.9 Chatter and User Info Page

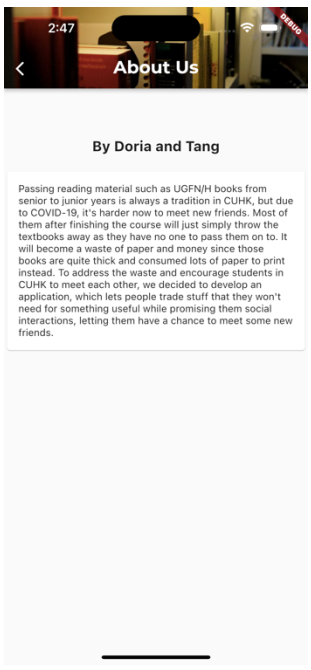
This page displays conversations between users. Users can send text messages as well as emojis. They can also send images from their photo gallery or take a picture using their camera to better describe something and enhance communication. Users can also click on the user icon to learn more about the person they are chatting with, such as the other books they offer.



If a trade exists between users, a bar with two buttons will be shown. The 'View Trade Bucket' button is a shortcut for users to quickly access and view the trade bucket, make a counteroffer, or simply accept and confirm the trade offer using the green button beside. More details about TI(Trade Interface) will be shown in 3.1.11.

Example of a user's profile:

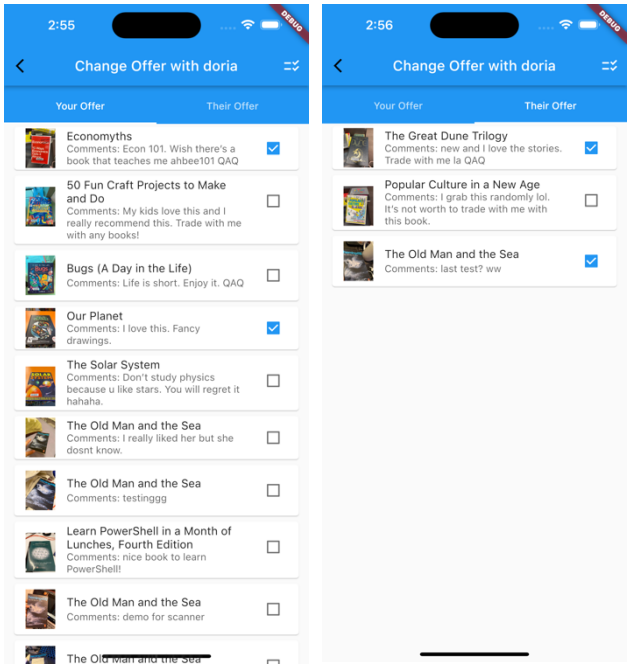




3.1.10 Setting Page

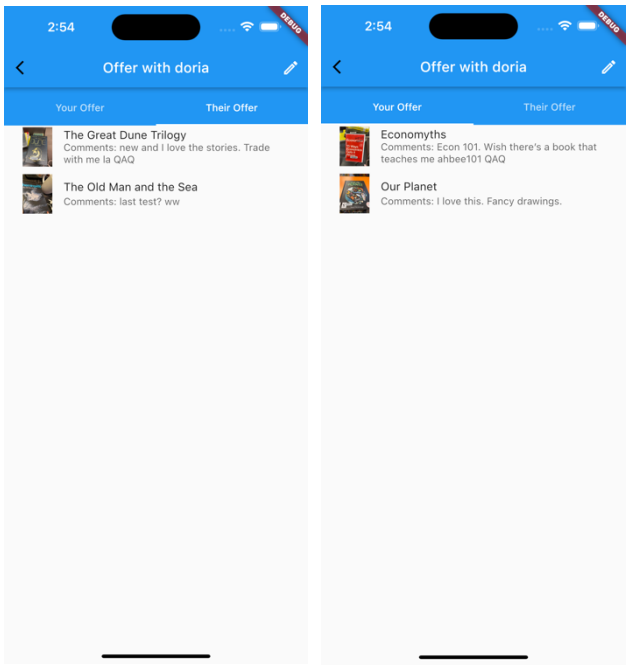
This page is for users to make modifications to their use of the application. Users can change their avatar, settings for what they want to share, enable dark theme, and know more about the project/developer through the "About Us" button. Users can also change their username and password and log off from their account on this page.

3.1.11 TI Page



TI stands for Trade Interface. Users can send a trade request to other users containing the books they want to give and receive from the other user. When the user wants to initiate a request, this interface will appear. For this page, the user will first need to select one or more items from their inventory for trading. Afterward, they can select what they want from the other user's inventory/trade list. After confirming, the trade request will be sent to the other user.

This page can also be reused when one party wants to change the trade offer by making a counteroffer. Once both users accept and confirm the trade offer, it will no longer be available for editing and will not be shown on the public search list.



Example of an offer :

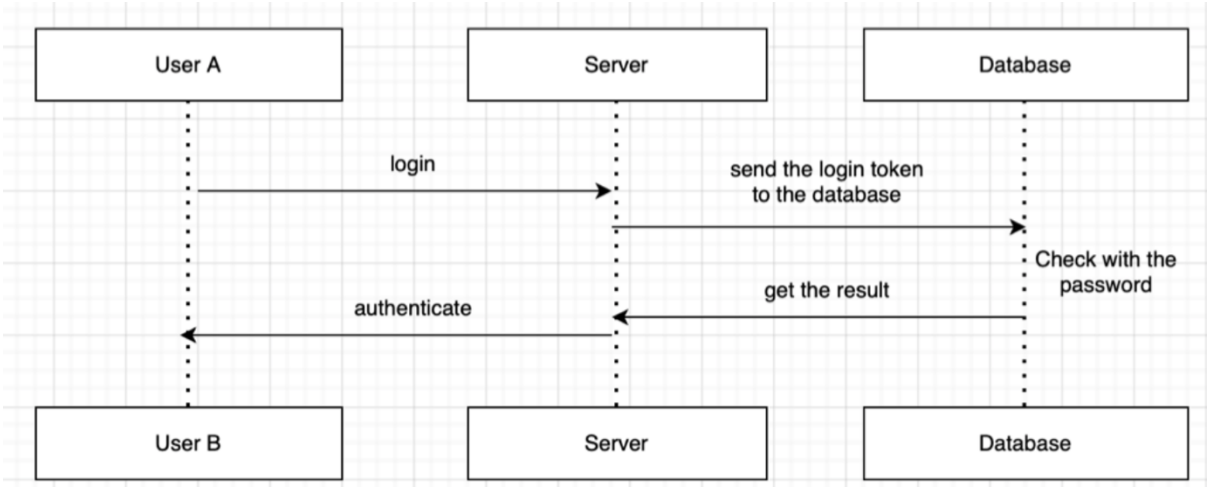
3.2 NodeJS as Middleware

Node.js is a widely used tool for developing server-side applications that perform CRUD operations on data stored in MongoDB. One of the reasons for its popularity is that it has an event-driven and non-blocking I/O model, which enables it to handle large amounts of data and high-traffic applications efficiently.

Another reason for using Node.js is because of security concerns. When developing an application, it is important not to store any credentials on the client side as the application could be reverse-engineered, and the API keys or any credential URL could be leaked. Therefore, it is best practice to store these sensitive pieces of information on the server side, which can be managed by ourselves. We can create APIs that allow authenticated users to call them instead of letting them perform CRUD operations on the database directly.

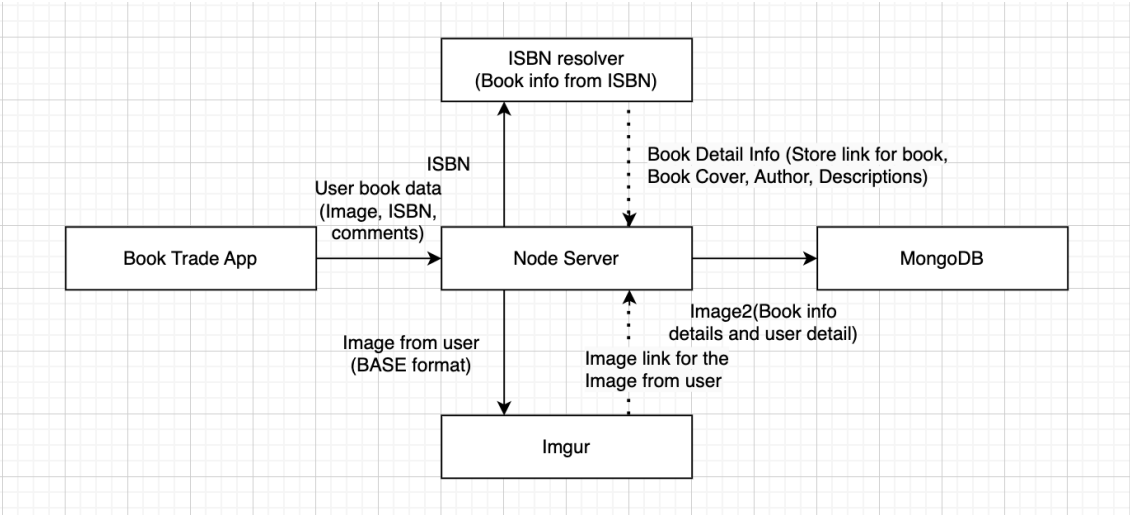
From now, we will introduce some of our main APIs that are crucial to our project.

3.2.1 Login API



The above diagram provides a simple way to demonstrate how data flows between the three main components for user registration and login to the app. Essentially, after the user enters their credentials, they are encrypted and hashed using the "bcryptjs" library. If the hashed string with salt matches the string in the database, the user can then be authorized and enter the application's home page.

3.2.2 Uploading API

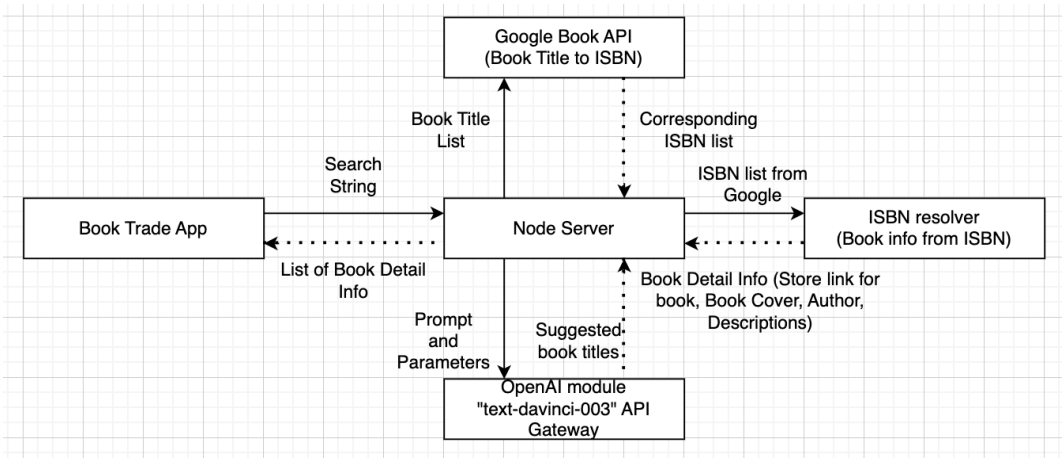


This API allows users to upload their books into our database. First, we retrieve the image shot by the user which will be set as the trade item's main image, and the ISBN that the user scanned through our ISBN scanner or input manually. Next, the data is sent to our Node.js server for further processing. In the server, the image is first sent to Imgur, which hosts the image on their server and returns a URL link for us to access it. This helps us save storage in our database and allows for easy access to the image.

After that, the ISBN provided by the user is used to collect more information about the book they uploaded. The node server will send the ISBN to the ISBN resolver for more info detail about the book, such as the book cover, link to the store for the book, etc.

After all the data has been refined, it is then stored in MongoDB. For more information about the content being stored in the database, please refer to the next chapter.

3.2.3 GrabGPT API



This API is designed to offer users a helpful list of book recommendations, up to three in total, based on a search string input. When a user inputs a search string, the information is sent to our Node.js server, which then calls the API to reach ChatGPT-3 with predefined prompts and fixed parameters.

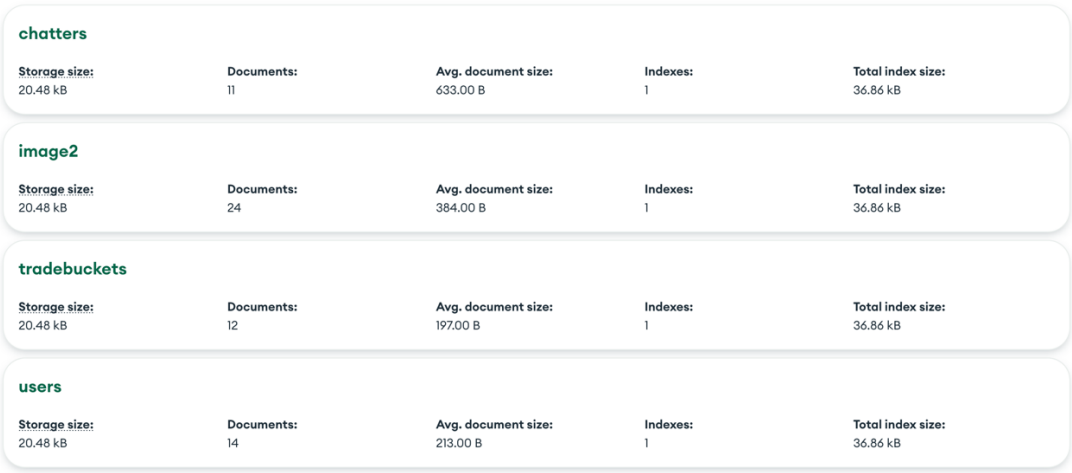
After the GPT generates a book title, we use the Google Book API to verify its existence. This is important because sometimes ChatGPT-3 will make up and suggest some non-existent books.

If the suggested book exists, the Google Book API will return a list of ISBN which is corresponded to the book title, and we then can send the list of ISBN to the ISBN resolver in order to gather additional information about the book.

After the node server received this data, the node server will send it back to the application and present it in a slideshow format for the user to select the books they prefer.

3.3 MongoDB as Database

We use MongoDB as our database since MongoDB is a non-relational database [4] which provides us the flexibility to implement the data flow we wanted. The image above shows the data we are storing in MongoDB, which can be divided into four main parts. The following image is the names of each database for storing data.



3.3.1 Users

This part stores the user credentials after they register an account and verifies the user's credentials upon login. It also stores the user's avatar (Imgur link) and username.

See the example below:

```
_id: ObjectId('63ad9c9e1aea9704abae64bf')
name: "doria"
email: "doria@gmail.com"
password: "$2a$08$rUMswEdhf6Xch30jz63XnOnJ.rWsI7aNsPlheU6q6XlySzt.qMCQC"
address: "https://i.imgur.com/k2XHNOy.jpg"
type: "user"
__v: 0
```

3.3.2 Image2

Image 2 is the place to store the book information that the user uploads. It contains the book's information and the uploader's information as well as the comment for the book from the uploader.

See example below:


```
_id: ObjectId('63b294c15bae82ed201fde0a')
name: "yjhKj12"
notself: "doria"
url: "https://i.imgur.com/yjhKj12.jpg"
dbISBN: "9781473224469"
delhash: "jQFrWUKgOR3YKeN"
comments: "new and I love the stories. Trade with me la QAQ"
googlelink: "http://books.google.com.hk/books?id=rPufswEACAAJ&dq=isbn:9781473224469…"
author: "Frank Herbert"
booktitle: "The Great Dune Trilogy"
__v: 0
```


3.3.3 Chatter

Chatter stores user chat records, such as the context they are chatting about, images, and timestamps. This self-defined format helps us to display it on the chat screen easier than using other packages or formats.

See the example below:


```
_id: ObjectId('63bfecf4d8df937aabad67d6')
self: "tangjaii"
notself: "doria"
randomhash: "52633"
lastdate: "2023-03-08"
▾ chatter: Array
  ▾ 0: Object
      dates: "2023-01-12"
  ▾ 1: Object
      user: "tangjaii"
      text: "Yo, I missu"
  ▸ 2: Object
  ▸ 3: Object
  ▸ 4: Object
  ▸ 5: Object
  ▸ 6: Object
  ▸ 7: Object
```

3.3.4 TradeBucket

The trade bucket is the place for storing the offers that each user made, such as which book they are willing to give and take. This is used in the trade bucket list for displaying the list of offered books between users.

See the example below:

4. Testing

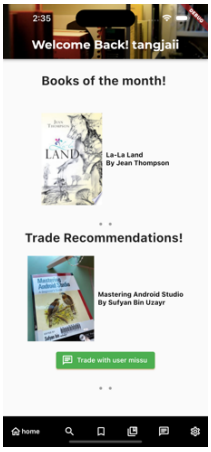
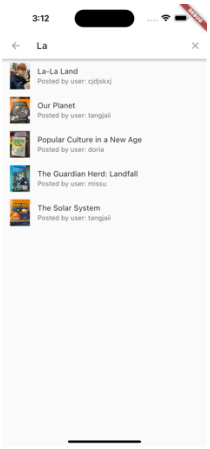
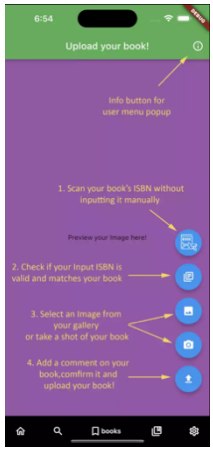
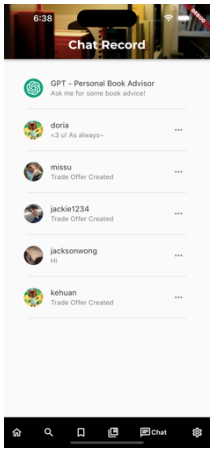
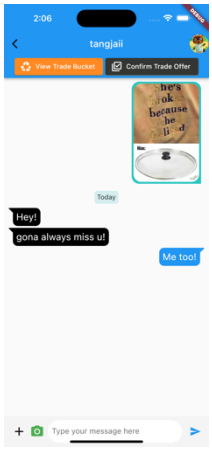
There will be four main parts to test out. The first will be UI/UX testing, which is mainly to test the front end of the application for the user. The next part will be functional testing, such as testing different buttons to see if they work on other devices and as intended. The third part will be middleware testing, which is to test the API on node server and see if it is on Desired State Configuration [5] for Database. The last part will be beta testing, where we will ask some of our friends to test the app and provide feedback to see if there are any improvements needed.

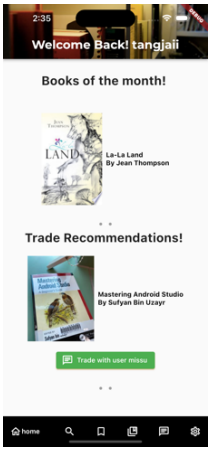
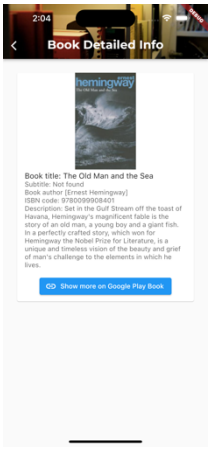
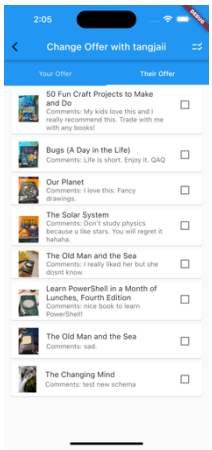
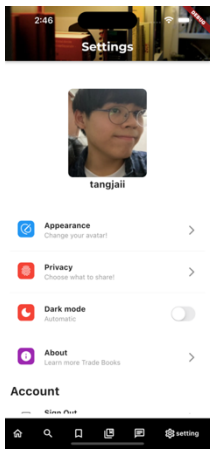
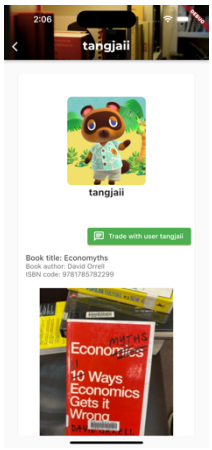
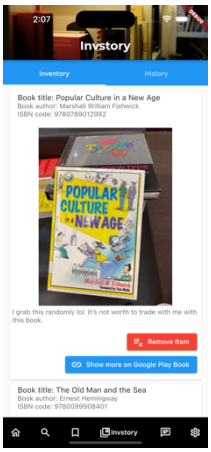
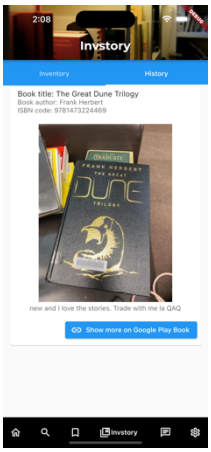
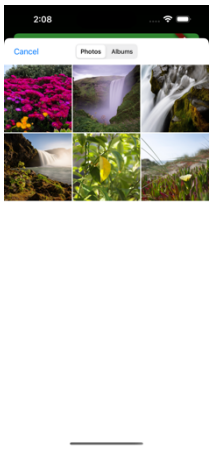
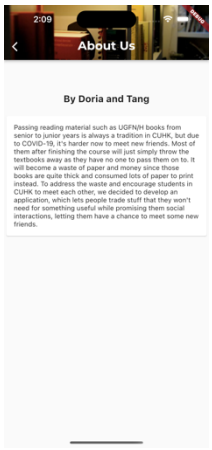
4.1 UI/UX testing

UI/UX testing involves evaluating the user interface and experience of an application to ensure it is user-friendly and easy to navigate. This type of testing is used to identify any usability issues that could impact user engagement and satisfaction. For this testing, we mainly test the in-app navigation and redirections, which is the process of moving from one screen or page to another within the app.

Testing in-app navigation involves checking that all buttons, links, and menus are in the correct locations and functioning as intended.

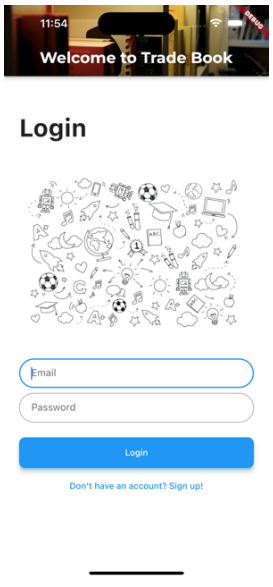
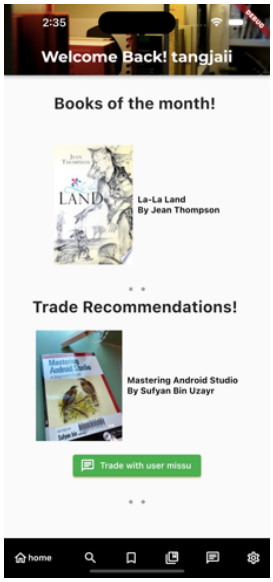
4.1.1 In-app Navigation Test

Home Page	Search Page	Upload Page	Chat Record Page	Chatter Page
				

Book Info Page	Url Prompt Page	Trade Bucket Page	Trade Select Page	Settings Page
				
User Info Page	**Inventory Page**	**History Page**	**Image Select Page**	**About Us Page**
				

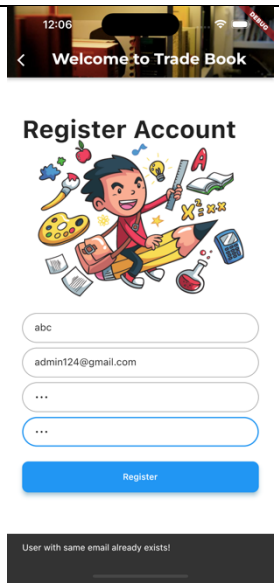
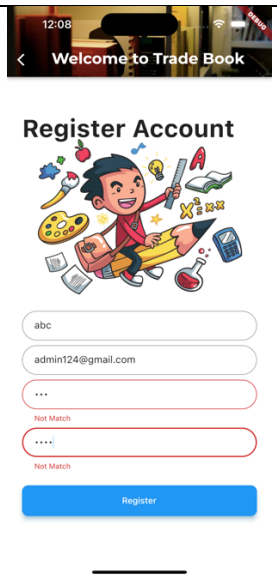
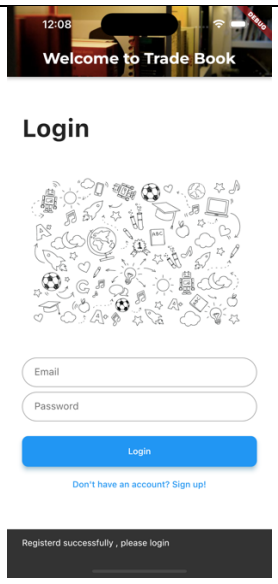
4.1.2 Login and Redirections

When a user enters their email and password, and there is a match in our database, they will be permitted to log in, and the application will navigate them to the home page. However, if they enter the wrong email or password, they will not be able to log in, and an error message will appear.

Login Page	User usefully logs in	A user enters a wrong email/did not register an account	User enters a wrong password
			

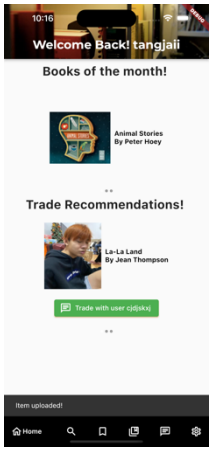
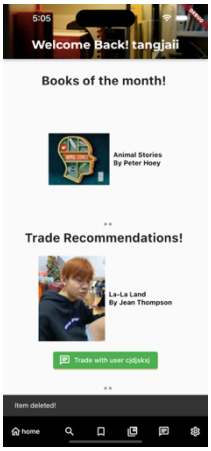
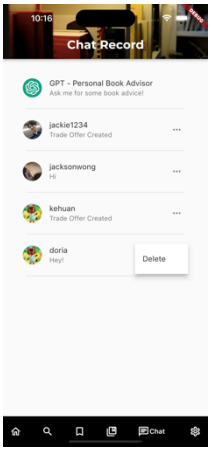
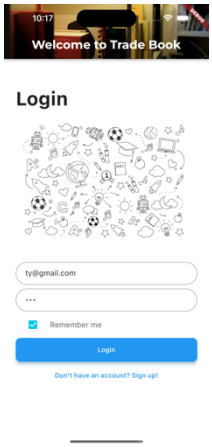
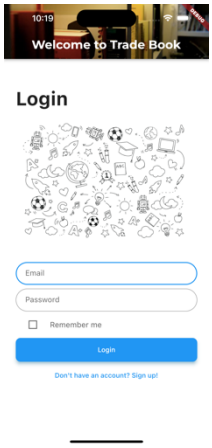
4.1.3 Registrations

This is where users can register for an account. To successfully register, users must input the required information. Once the information is entered correctly, they can register for an account. After successful registration, the application will navigate the user to login page to log in.

Email already taken	Password mismatched	Register success
		

4.2 Functional Testing

Functional testing involves testing buttons such as the upload button, delete button, logout button, and other main function buttons to ensure that they are working as intended. The purpose of functional testing is to verify that the system's functionality aligns with the requirements. Testing the various buttons to ensure that the system is functioning as expected. See examples below:

Upload a book	Delete a book	Delete a chat	Remember me	Logout
				

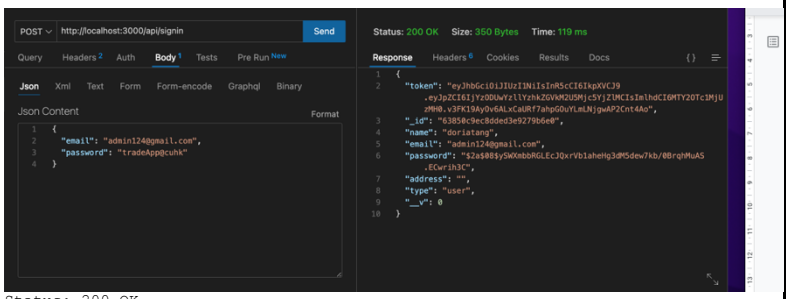
4.3 Basic Node APIs test

We will be testing the different APIs that we have built for the application to ensure that they are functioning correctly. This will involve testing APIs such as the registration, login, and book information retrieval APIs, among others. By thoroughly testing these APIs, we can identify and address any issues before the app is released to the public, ensuring all client side functions can get the data they need from our node server.

4.3.1 Register API test:

The Backend server will send a POST request to the database and see if there is a match inside the database. It will then return a response.

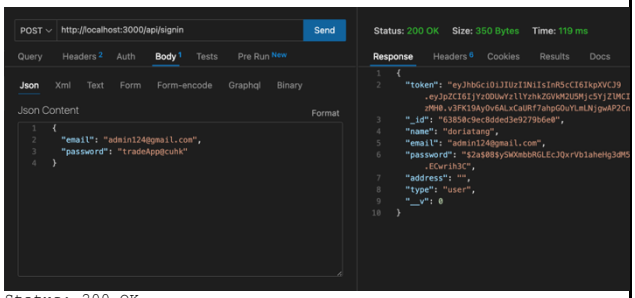
JSON Request	JSON Response	Testing Result

<pre>{
  "email":
"admin124@
gmail.com"
,
  "passwor
d":
"tradeApp@
cuhk"
}</pre>
	<pre>{
	  "name": "doriatang",
	  "email":
	"admin124@gmail.com",
	  "password":
	"$2a$08$ySWXmbbRGLEcJQxr
	Vb1aheHg3dM5dew7kb/0Brqh
	MuAS.ECwrih3C",
	  "address": "",
	  "type": "user",
	  "__v": 0
	}</pre>
		
		Status: 200 OK

4.3.2 Login API test:

The Backend server will send a POST request to the database and see if there is a match inside the database. It will then return a response.

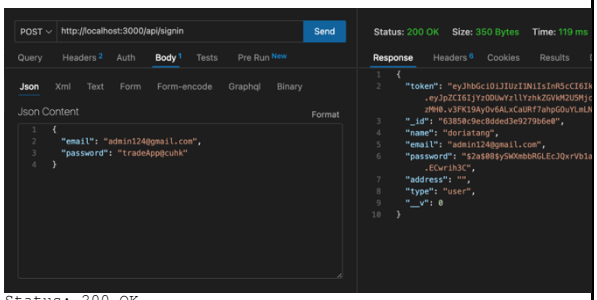
JSON Request	JSON Response	Testing Result

<pre>{
  "email":
"admin124@
gmail.com"
,
  "passwor
d":
"tradeApp@
cuhk"
}</pre>
	<pre>{
	  "token":
	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
	JpZCI6IjYzODUwYzllYzhkZGVkM2U5Mjc5YjZlMC
	IsImlhdCI6MTY2OTc1MjUzMH0.v3FK19AyOv6ALx
	CaURf7ahpGOuYLmLNjgwAP2Cnt4Ao",
	  "_id": "63850c9ec8dded3e9279b6e0",
	  "name": "doriatang",
	  "email": "admin124@gmail.com",
	  "password":
	"$2a$08$ySWXmbbRGLEcJQxrVb1aheHg3dM5dew7
	kb/0BrqhMuAS.ECwrih3C",
	  "address": "",
	  "type": "user",
	  "__v": 0 }</pre>
		
		Status: 200 OK

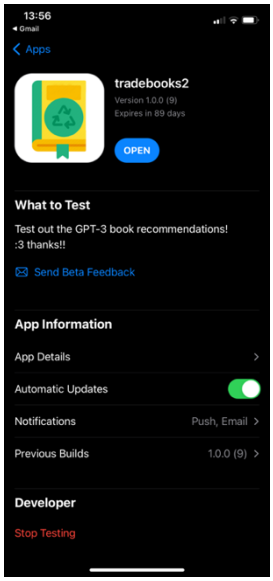
4.3.3 Register API test:

The Backend server will send a POST request to the database and see if there is a match inside the database. It will then return a response.

JSON Request	JSON Response	Testing Result

<pre>{
  "email": "admin124@gmail.com",
  "password": "tradeApp@cuhk"
}</pre>
	<pre>{
	  "name": "doriatang",
	  "email": "admin124@gmail.com",
	  "password": "$2a$08$ySWXmbbRGLEcJQxrVb1aheHg3dM5dew7kb/0BrqhMuAS.ECwrih3C",
	  "address": "",
	  "type": "user",
	  "__v": 0
	}</pre>
		
		Status: 200 OK

4.4 Beta Test



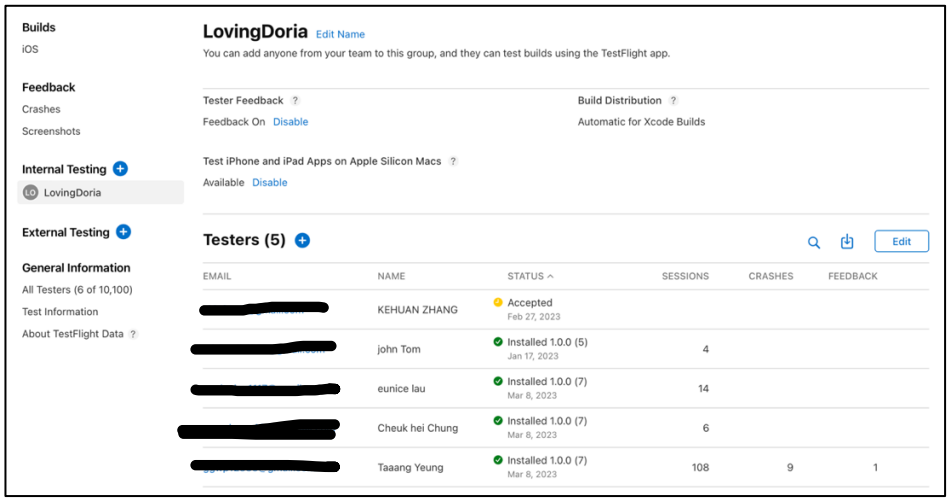
For the last part of testing, we will distribute the application to our friends at CUHK to beta test it and provide feedback for using the application and see how it compares to other similar applications like Carousell, and whether they have met any new friends after using our application. This will help us determine if our application is a successful and promising product.

After receiving feedback from our users, we have made some improvements to the UI to enhance the overall user experience. One of the changes we made was the addition of an info button, located in the top right corner of the screen, which provides users with more information about how to use the application such as uploading a book.

Another feature we added is called "Read with GPT." This feature allows users to get book recommendations from GPT after entering a search string.

We are excited to introduce these new features to them after we implanted them and look forward to hearing more feedback from our users as we continue to develop and improve our application, even after the end of this project!

The image below shows our current tester for the app:



5. Conclusion

In this project, We have developed a book exchange platform that let CUHK students trade books instead of throwing them away and becoming waste. With features like user registration, book searching, uploading, trading, and instant messaging, the app provides a unique experience for book trading. Features like “Read with GPT” aid users to find a book that fits their interest, which let users discover some books that they might like.

Not only does our app promote a green campus [6] and reduce waste, it also creates opportunities for social interactions. Users can interact, discuss books, coordinate trades, and build relationships with like-minded individuals and trade books with them. Sharing books allow people to discover little-known titles or special editions they may never find otherwise. Reusing books through trading can also raise awareness of consumerism and encourages sustainable habits to prevent waste.

Overall this project is a success and we have met our goals. We have developed and deployed a cross-platform book trading App on both Android and iOS. The implemented social interaction features like chat rooms, trade functions and UI’s for users to trade books can prevent book waste. Providing book recommendations to enhance the user experience and bring convenience to them. The project has achieved all the set objectives.

6. Future Developments and Improvements

6.1.1 Improvements with physical devices on the network configurations

As we are testing and deploying our codes on XCode using the built-in simulator, we do not need to manually configure the simulated device since it is already in developer mode. However, when we tried it on our physical phone, we encountered some issues regarding the network and the application was unable to send or receive any requests or responses from the server. To address this issue, we may need to configure the iOS configuration files so that we can have permission to communicate with the server on our phone. During the deployment of Android, we encountered the same issue. After configuring the Android manifest system file and setting up a DNS address from Google (8.8.8.8), the problem was resolved. We will take notes from here and be reminded of these kinds of issues during the development of the web application for Book Trade App.

6.1.2 Improvements in how we implement the codes

Since we were mainly focused on the application architecture, many of our codes were messy which just fulfilled the data flow and basic functions. For example, we were sending data and uploading images directly from the client side, which could raise some security concerns. As a result of the unorganized codes, we did not have a technical specification for the codes that we were implementing, and many of the codes were not reusable. To make the codes clearer and more reusable, we will set technical specifications for the codes and APIs that we are using, to organize our application in a more detailed and well-structured way.

6.1.3 Improvements in Performance

Currently, we have included all the API calls in our node server to save, host, and load data. However, we would like to migrate some of those calls to the client side for non-API key needed functions. For example, looking up an ISBN has a public API, and no keys or credentials are required. If we can migrate those functions to the client side, it will improve the performance of the app by reducing the round trip between the node server and client, resulting in faster loading for the book data.

6.1.4 Improvements in Documentation

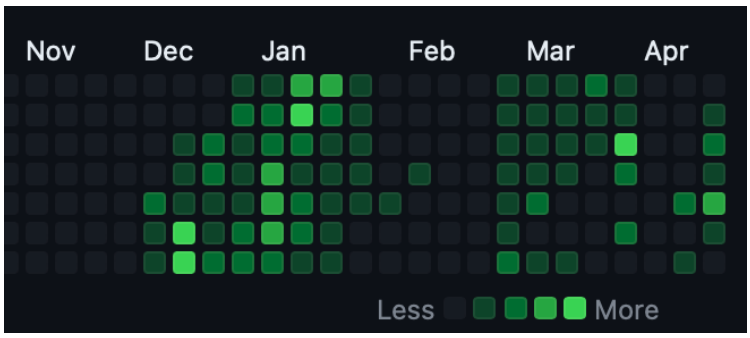
During the app development process, most of the code was written in "noodle" and lacked comments, making it difficult to read and understand for other developers. To improve readability, we should add more comments and use understandable variable names. We should also create a document specifically for the API functions that explains what they do, what parameters they take, and other relevant information after creating the API. This will help developers or feature us to better understand the code before and make future modifications and updates more efficient and simple.

6.1.5 Improvements in Project Management and Self-Reflection

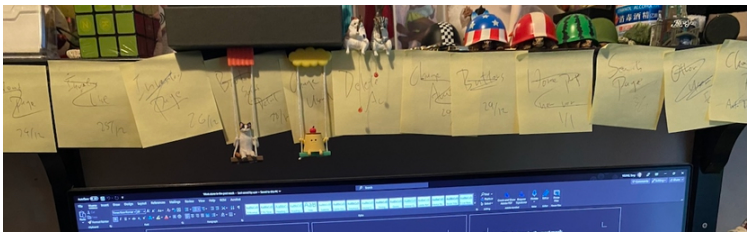
The slow progress of the development of the Book Trade Application during the first semester was not acceptable and it was mainly my fault. I let my own irresponsibility get the best of me, causing significant delays and setbacks that impacted the entire project. It was a difficult realization to come to, but I knew that I had let myself and my team down which eventually set both me and my partner into separate ways.

After recognizing my mistakes, I felt a strong sense of responsibility to make things right. I knew that I needed to take action to improve my

productivity to ensure that the project could move forward and be completed before the end of the year. I started by setting realistic goals and deadlines for myself and working hard to meet them. I also took the time to reflect on my behavior and make changes to my approach to work.



Git Commits



Schedule Wall

Through this project, I learned the importance of taking ownership of my responsibilities and not relying on others to do my work for me, also to always plan and schedule ahead for projects or assignments. I would like to thank my partner for her contributions and efforts during the first semester. It was an honor to meet her and to work alongside her.

This was indeed a lesson to learn at a big cost and a hard way, but this will be an experience that I have learned from, reflected on, and will remember forever.

6.2 Future Directions

6.2.1 Expanding to other categories

We hope that our application will not just stop with books, but expand to other categories such as electronics, housewares, and other items that people commonly throw away when they are no longer needed. For

example, many bed sheets or refrigerators are thrown away after people finish their undergraduate studies and no longer live in the dormitory, and our application hopes to address this type of waste.

6.2.2 Subscriptions

We hope to include subscriptions in our application since we want it to be self-sustainable. Including subscriptions can help pay for the fees for renting cloud services or other maintenance fees. Users can enjoy an ad-free experience and up to 20 item slots for trading upon subscribing to the monthly subscription.

6.2.3 Implementing a web application for Book Trade App

As we have already implemented the Book Trade app, which is available on both iOS and Android, we would also like to develop a web app that provides the same services hosted on websites. The web app will be easily accessible to most people since there is no limitation of using what operating system for a web app, which can potentially increase the number of users of our app and the volume of books that can be traded. Providing more options for users will enhance their experience.

6.2.4 Extra features

We would like to add more features to help users discover more about books and people while using our app. One such feature is a chat forum similar to Reddit, where users can start a thread and others can join the discussion. Users can share feedback, thoughts, and current book trends with each other, creating a more engaging and interactive experience.

6.2.5 Promotions

Currently, we are promoting our app only to our friends, which makes it difficult to reach a large number of users. To increase the app's exposure, we plan to leverage social media platforms such as Instagram and Facebook. We can also ask the admin for the page CUHK Secrets to help us promote the app once it is refined and ready for publication.

We have already created an Apple developer account, and once the app is ready, we can publish it on the App Store. This should help increase the number of users and promote our app to a wider audience.

7. Final Consolidated Logbook

Date	Work done during the week	Challenges/Problems	Task planned for next week
30/1	Finished the Inventory system and the upload function.	Using the image hosting service from Imgur.	Setup AWS cloud server and network for connecting to the app.
6/2	Finished setting up the AWS cloud server and the network. Got the apple developer account as well.	Setting up a AWS cloud server at a closer distance (Singapore) and getting the developer account.	Implement the Search function and a list view to browse books.
13/2	Finished the Search and Browse page for users to look for a book.	Implementing the search function in the Search page.	Finish up the Search page and start developing the Chatter.
20/2	Done with the Search page and the initiate stage of developing the Chatter.	Finding a nice UI package for chat dialog boxes and implementing the structure of storing messages.	Finishing the Chatter and get the first message to be sent out.
27/2	Finished the Chatter and users now can chat with each other.	The whole process of implementing the Chatter.	Deploying the app to TestFlight and test it on our iPhones.
6/3	Deployed the app on Testflight.	The process of generating a successful build to archive and upload it to TestFlight.	Try to get some feedback from other users and to add image in Chatter.
13/3	Implementing the TradeBucket.	The structure of storing book lists for users and the functions to modify the list.	Finish up the TradeBucket.
20/3	Finished TradeBucket.	To have 2 tabs in the TradeBucket page for a pair of users.	Improve UIs and the UX for better experience.
27/3	Improved buttons and SnackBars for better instructions.	To understand my unstructured noddle codes and UI overflow bugs.	Ask a friend and get some user feedback.
3/4	Added info buttons from feedback for better instructions.	To find a best way that user can interact with.	Deploy it on an Android platform to test out the “cross-platform application”
10/4	Successfully deployed on Android Google Pixel Series.	Some codes and UI placement need to be rewritten to fit the screens.	Fix the UI overflow issue and start on the final report.

8. Personal Logbook

This is the development log for the extra feature “Read with GPT” after 10/4.

Date	Work done during the week	Challenges/Problems	Task planned for next day/days
12/4	Fixed the UI for the overflow problem.	To find the right function to display the list.	Plans for including ChatGPT in the app.
14/4	Planned what to add in the app for using the ChatGPT.	Getting the use of ChatGPT (Hong Kong is forbidden to use GPT since “Hello Hong Kong! ”)	Find a way to use ChatGPT.
16/4	Found a way to get the API Key for using openAI resources.	Getting a number from foreign countries for verification.	Start to study and use the openAI API.
17/4	Tried the first call on using the openAI API.	Not much, just need to spend time reading the documents.	Develop a page for displaying the book recommendation from using GPT
18/4	Crated a simple page to display results.		Design a prompt that is efficient in using tokens and giving an accurate result.
19/4	Finish setting up the prompt.	It took me a day for making a good prompt since GPT is always providing fake ISBN numbers with the corresponding book. **Truth is hard, GPT lies!**	Finish up all the UIs and create a node API for calling the book recommendations.
20/4	Finished the UI and the node API.	After having the suggested book title from GPT, I still need to get a correct ISBN with the title. I then used the GoogleBook API for retrieving the correct ISBN and feeding it into an OpenBook API for getting the results of the corresponding book from multiple sources.	Finish up the whole project and the final report :D

9. Appendix for Package used

9.1 Appendix for Flutter Package used

9.1.1 cupertino_icons: ^1.0.2

This asset repo contains the default set of icon assets used by Flutter's Cupertino widgets.

9.1.2 google_nav_bar: ^5.0.6

A modern google style nav bar for flutter.

9.1.3 mobile_scanner: ^2.0.0

A universal barcode and QR code scanner for Flutter based on MLKit. Uses CameraX on Android, AVFoundation on iOS and Apple Vision & AVFoundation on macOS.

9.1.4 http: ^0.13.5

A composable, Future-based library for making HTTP requests. This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform and supports mobile, desktop, and browser.

9.1.5 carousel_slider: ^4.1.1

A carousel slider widget.

9.1.6 flutter_image_slideshow: ^0.1.4

A simple image slideshow widget. Mainly intended for image widgets, but other widgets can also be used.

9.1.7 file_picker: ^5.2.2

A package that allows you to use the native file explorer to pick single or multiple files, with extensions filtering support.

9.1.8 provider: ^6.0.4

A wrapper around InheritedWidget to make them easier to use and more reusable.

9.1.9 shared_preferences: ^2.0.15

Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.). Data may be persisted to disk asynchronously, and no guarantee that writes will be persisted to disk after returning, so this plugin must not be used for storing critical data.

9.1.10 google_fonts: ^3.0.1

A Flutter package to use fonts from fonts.google.com.

9.2 Appendix for Node Packages used

9.2.1 "bcryptjs": "^2.4.3"

Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power. ([see](http://en.wikipedia.org/wiki/Bcrypt))

9.2.2 "express": "^4.18.2"

The fast, unopinionated, minimalist web framework for Node.js.

9.2.3 "http": "^0.0.1-security"

An npm package that holds a spot.

9.2.4 "jsonwebtoken":"^8.5.1"

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

9.2.5 "mongoose":"^6.7.2"

Mongoose provides a straightforward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks, and more, out-of-the-box.

9.2.6 "node-isbn": "^1.6.1"

Find books by ISBN.

10. Reference

[1] Fong, Ben Y. F. et al. “Relationships Between Physical and Social Behavioural Changes and the Mental Status of Homebound Residents in Hong Kong During the Covid-19 Pandemic.” International journal of environmental research and public health 17.18 (2020): 1–18. Web.

[2] Yaser, Abu Zahrim. Green Engineering for Campus Sustainability. Ed. Abu Zahrim. Yaser. 1st ed. 2020. Singapore: Springer Singapore, 2020. Print.

[3] M. D. Hopp, M. Händel, S. Bedenlier, M. Glaeser-Zikuda, R. Kammerl, B. Kopp, and A. Ziegler, “The structure of social networks and its link to higher education students’ socio-emotional loneliness during covid-19,” *Frontiers in Psychology*, vol. 12, 2022.

[4] Medina, Juan Miguel, Ignacio J. Blanco, and Olga Pons. “A Fuzzy Database Engine for mongoDB.” International journal of intelligent systems 37.9 (2022): 5691–5724. Web.

[5] Sdwheeler, “Get started with desired state configuration (DSC) for Windows - PowerShell,” PowerShell | Microsoft Learn. [Online]. Available: https://learn.microsoft.com/en-us/powershell/dsc/getting-started/wingettingstarted?view=dsc-1.1.

[6] Sugiarto A, Lee CW, Huruta AD. “A Systematic Review of the Sustainable Campus Concept.” Behav Sci (Basel). (2022): 3.2. Learning Tool Dimension

11. Veriguide Report

The Chinese University of Hong Kong
Academic Honesty Declaration Statement

Submission Details

Student Name	YEUNG Tang (s1155144676)		
Year and Term	2022-2023 Term 2		
Course	IERG-4999-SJ01 Final Year Project II		
Assignment Marker	Professor ZHANG Kehuan		
Submitted File Name	1155144676_FYP_Report.pdf		
Submission Type	Individual		
Assignment Number	3	**Due Date (provided by student)**	2023-04-22
Submission Reference Number	3657922	**Submission Time**	2023-04-22 17:00:59

Agreement and Declaration on Student's Work Submitted to VeriGuide

VeriGuide is intended to help the University to assure that works submitted by students as part of course requirement are original, and that students receive the proper recognition and grades for doing so. The student, in submitting his/her work ("this Work") to VeriGuide, warrants that he/she is the lawful owner of the copyright of this Work. The student hereby grants a worldwide irrevocable non-exclusive perpetual licence in respect of the copyright in this Work to the University. The University will use this Work for the following purposes.

(a) Checking that this Work is original
The University needs to establish with reasonable confidence that this Work is original, before this Work can be marked or graded. For this purpose, VeriGuide will produce comparison reports showing any apparent similarities between this Work and other works, in order to provide data for teachers to decide, in the context of the particular subjects, course and assignment. In addition, the Work may be investigated by AI content detection software to determine originality. However, any such reports that show the author's identity will only be made available to teachers, administrators and relevant committees in the University with a legitimate responsibility for marking, grading, examining, degree and other awards, quality assurance, and where necessary, for student discipline.

(b) Anonymous archive for reference in checking that future works submitted by other students of the University are original
The University will store this Work anonymously in an archive, to serve as one of the bases for comparison with future works submitted by other students of the University, in order to establish that the latter are original. For this purpose, every effort will be made to ensure this Work will be stored in a manner that would not reveal the author's identity, and that in exhibiting any comparison with other work, only relevant sentences/ parts of this Work with apparent similarities will be cited. In order to help the University to achieve anonymity, this Work submitted should not contain any reference to the student's name or identity except in designated places on the front page of this Work (which will allow this information to be removed before archival).

(c) Research and statistical reports
The University will also use the material for research on the methodology of textual comparisons and evaluations, on teaching and learning, and for the compilation of statistical reports. For this purpose, only the anonymously archived material will be used, so that student identity is not revealed.

I confirm that the above submission details are correct. I am submitting the assignment for:

[X] an individual project.

I have read the above and in submitting this Work fully agree to all the terms. I declare that: (i) the assignment here submitted is original except for source material explicitly acknowledged; (ii) the piece of work, or a part of the piece of work has not been submitted for more than one purpose (e.g. to satisfy the requirements in two different courses) without declaration; and (iii) the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I also acknowledge that I am aware of the University's policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website http://www.cuhk.edu.hk/policy/academichonesty/.

I declare that I have not distributed/ shared/ copied any teaching materials without the consent of the course teacher(s) to gain unfair academic advantage in the assignment/ course.

I declare that I have read and understood the University's policy on the use of AI for academic work. I confirm that I have complied with the instructions given by my teacher regarding the use of AI tools for this assignment and consent to the use of AI content detection software to review my submission.

I also understand that assignments without a properly signed declaration by the student concerned will not be graded by the teacher(s)

Signature (YEUNG Tang, s1155144676)

22/4

Date

Instruction for Submitting Hard Copy / Soft Copy of the Assignment

This signed declaration statement should be attached to the hard copy assignment or submission to the course teacher, according to the instructions as stipulated by the course teacher. If you are required to submit your assignment in soft copy only, please print out a copy of this signed declaration statement and hand it in separately to your course teacher.

Page: 1