Team Name: Deep Optimum

Team Member UNI:
Beom Joon Baek - bb2763
Bennington Li - wl2750
Yifan Zang - yz3781
Zhuoxuan Li - zl2890

Part 0:

Date: November 4th, 2020

The teaching staff will review the preliminary project proposals, and then a specific IA mentor will be assigned to your team. You should meet with your IA mentor to discuss your proposal before submitting the revision, and in any case your revised proposal will not be graded until after you have met with your IA mentor.

Submit the date of your meeting with your IA mentor (note this date must be in the past, not the future).

Part 1:

Take into consideration feedback from your IA mentor or other members of the teaching staff. Revise accordingly answers all five of the numbered questions. If you disagree with the feedback, explain why.

1. What will your project do?

Initially, we just wanted to create a simple application that allows students to purchase/sell textbooks online. After discussing with Shirish, we realized that the original proposal was too simple and lacked a lot of functionalities and we should add more features to it. We switched the apple/facebook login api to gmail login api since we are specifically targeting Columbia students, whose UNIs are associated with Gmail. We decided to use the Google maps API that allows us to locate our users and the listing and give users an opportunity to view items that are within a certain radius. We also added an email api that will automatically send out emails to both parties when an order has been placed via the platform. We were also planning to add a shipping feature to the project during the discussion, but then we decided not to. This is mainly because we are targeting local buyers/sellers and focusing on local transactions and so there is not much need for shipping. So the following is a summary about our project:

A marketplace application that offers great convenience to Columbia students who want to purchase textbooks. It is the simplest, most trusted way to buy and sell textbooks locally. **We empower students to connect and prosper**. Users can search textbooks by either **keywords** or by **categories**, browse textbooks available on the platform, and post want-to-buy requests. Users can also post their textbooks for sales. Once the user finds something of his/her interest, the user can contact the seller via the chat feature instantly and securely through the app to finalize the time and location to meet. Email confirmations will be sent to both parties. Transactions completed on the platform are secure and protected. Both parties have to agree to complete the transaction during the meetup. Buyers must scan the QR code on the seller's device in order to complete the transaction, and the seller will not receive the funds unless the QR code has been scanned. This extra step provides protection for smooth and secure transactions for both parties.

2. Who or what will be its users? Your project must impose registration, authenticated login and timeout/explicit logout, so your answer should say something about this.

Columbia students will be the users. All users must be registered in order to use the website. Users are required to register with their Columbia email account via Google login authentication. If a user is not active for more than 20 minutes, this user's session will end.

3. Your project must be demoable, but does not need a GUI if there's a command line console or some other way to demonstrate. (All demos must be entirely online, there will be no in-person demos.) What do you think you'll be able to show in your demo?

We will be able to present our project via a website. All the features we are planning to add will also be demoable on the website. We will demo:

1. The layout by browsing through the website,
2. The search functionality by typing keywords or categories into the search bar.
3. The login functionality by login with our Columbia email account
4. The chat functionality by starting a chat between two of us
5. The posting functionality by creating and uploading a new post.
6. The transaction functionality by purchasing a posted textbook and showing the balance change in paypal accounts. Scanning QR code could be a little inconvenient in an online demo, but it should be viable to scan through the screen.

4. Your project must store and retrieve some application data persistently (e.g., using a database or key-value store, not just a file), such that when your application terminates and starts up again some arbitrary time later, the data is still there and used for later processing. What kind of data do you plan to store?

We are planning to store user information in SQL. We are going to use SQLAlchemy. We will create four tables; one table for users logistics, one table for all transactions, one table for all listings, and one table for all want-to-buy-requests. User logistics table contains user email, username, userID, user's location, listing history, purchase history, request history. The listing table contains the listing's categories, date, and listing location. The want-to-buy-requests table contains request's categories, date, and request location.

5. Your project must leverage some publicly available API beyond those that "come with" the platform; it is acceptable to use an API for external data retrieval instead of to call a library or service.. The API does not need to be a REST API. There are many public APIs linked at https://github.com/public-apis/public-apis (Links to an external site.) and

https://github.com/n0shake/Public-APIs (Links to an external site.). What API do you plan to use and what will you use it for?

We are planning to use these external APIs.
1. Stream Chat API: We will use stream chat api for communication between sellers and buyers.
2. Emailjs API: We will use emailjs api for sending email confirmation such as account creation, password reset, order placement.
3. Google Maps API: We will use Google Maps API in two instances:
    1. When a SELLER lists a textbook, they HAVE to upload the location they are situated. They will pin their current location on Google Maps and their location will be associated with the textbook.
    2. When a BUYER wants to buy a book, they can have a list of 5 closest SELLERS from the BUYER's location. The BUYER will pin their current location on Google Maps, and based on that location, it will show the list of 5 closest SELLERS.

4. Paypal API: We will use Paypal API for micro-transaction between BUYER and SELLER.
5. Google Sign In API: We will use Google Sign in API to verify the account of the BUYER or SELLER.
6. OpenCV API: We will use this for QR code generation and reading. Users will use QR code to verify the transaction.

Part 2:

Take into consideration feedback from your IA mentor or other members of the teaching staff and revise accordingly. If you disagree with the feedback, explain why. **Write three to five user stories for your proposed application**, constituting a Minimal Viable Product (MVP); generic functionality like registration, login/logout and help, should not be included among these user stories. That is, your application should do at least three application-specific things. Use the format

< label >: As a < type of user >, I want < some goal > so that < some reason >.

My conditions of satisfaction are < list of common cases and special cases that must work >.

The type of user (role) does not need to be human. You may optionally include a wishlist of additional user stories to add if time permits. Keep in mind that the type of user, the goal, the reason (if applicable within the system), all the common cases and all the special cases must be testable and demoable.

We have modified the following user stories according to IA's feedback. After edits, the following user stories are more specific

(1) As a **student buyer**, I **want** to be able to purchase cheap and used textbooks locally for my classes at Columbia **so that** I do not have to pay extremely high prices on Amazon or Ebay and wait for a week..

My conditions of satisfaction are:

- I can search textbooks by keywords or by category.
- If there are no textbooks I want to find, I can post a request for that textbook.
- I can browse textbooks by category or by distance.
- If there are textbooks I want to buy, I want to be able to contact the seller.
- I want to be able to see the condition of the books via pictures/videos.
- If I decide to purchase the textbook, I want to have buyer protection; meaning I want my money protected. If the seller did not fulfill the agreement, I want to be refunded.
- If the transaction is successful, I will get email notification just for the record.

(2) As a **student seller,** I **want** to be able to post my textbooks that I don't need anymore and sell them locally **so that** I can make the best use of them.

My conditions of satisfaction are:

- I can categorize the textbooks I am selling.
- I can give text-based descriptions along with pictures of the textbooks that I am selling.
- I can set a price for the item.
- I can chat with buyers and set up time and location for local meetups.

(3) As a **student seller**, I **want** to be able to receive my payments securely and promptly after the transaction of the textbook so that I can get paid.

My conditions of satisfaction are:

- After chatting with the buyer to coordinate a time to meet up, I expect the buyer to show up at the location.
- If the buyer does not show up at the location, I can cancel the transaction and have my textbook re-listed on the website.
- If both the buyer and the seller meet in-person and if the buyer agree to the transaction after seeing the textbook in-person, I scan the buyer's QR code. That will finalize the transaction and I get paid for the textbook.
- If the buyer decides that the physical condition of the book is not good or that the book is somehow wrong, the buyer can cancel the transaction. That will have the textbook re-listed on the website.
- If the transaction is successful, I will get an email notification of the successful transaction.

(4) As an **admin**, I want to be able to **get the total amount of transactions** on the website so that I can use it to track my revenue from the website.

My conditions of satisfaction are:

- I can see an aggregate of all the previous transactions on the website.
- I can see the monthly revenue of the website. If the monthly revenue is down from the previous month, it shows the figure in red. If the monthly revenue is up, then it shows in green.

Part 3:

Take into consideration feedback from your IA mentor or other members of the teaching staff. Revise accordingly how you will conduct **acceptance testing** on your project. If you disagree with the feedback, explain why. Every MVP user story must be associated with a plan for user-level testing. The test plan should address all its common cases and all its special cases. Discuss sample inputs the user or client would enter and the results expected for the corresponding test to pass vs. fail. Note inputs might come from files, network, devices, etc., not necessarily from a GUI or command line, and results might involve changes in application state, files, outgoing network traffics, control of devices, etc., not necessarily outputs via a GUI or command line. You may optionally discuss testing plans for your wishlist additional user stories, if any.

1. Student buyer acceptance testing

Common cases:

(a) User inputs the keywords of the textbook in the text field, or chooses predefined categories for textbook selections.

  (i) If found, then a list of relevant items should be returned and shown on the website. Assert if the same list of relevant items from the database query appears on the website.

  (ii) If not found, it should return a message "Nothing is found" with the corresponding error code. Assert the returned message.

  (iii) Each item in the return list must include at least one photo, and the user should be able to look at the photos/videos of the item. Assert if any of this is present on the website.

  (iv) The user should be able to contact the seller and chat with the seller for the item. Assert if a message sent from one end appears on both ends.

  (v) The user can pay via paypal. The test fails if the user cannot pay via paypal.

  (vi) After goods have been exchanged, the user should be able to provide a QR code for the seller to scan. Assert if the QR code is present.

(vii) Once the QR code has been scanned, the transaction will be marked complete and funds will be transferred to the seller. We can test whether the funds will show up in the seller's account after the QR code has been scanned. The test fails if this cannot be done.

(viii) After the transaction, the item should be marked “unavailable” and the post should be archived. Assert if this is true.

(b) User shares the location via the browser or enters an address and chooses to view the “textbook” nearby.

(i) If there are textbooks available nearby, then a list of relevant items should be returned and shown on the website. We can assert if the list returned by the database is present on the website.

(ii) If not found, it should return a message “Nothing is found” with the corresponding error code. We can assert the returned message.

(iii) The rest is the same as part a.

(c) User inputs the name of the item but cannot find the item.

(i) The user will see a message “Nothing is found.” If this cannot be done, the test fails. We can assert the returned message.

(ii) The user can post a “Want to buy” request via a “Want to buy” button on the website. If this cannot be done, the test fails.

(iii) The user categorizes the textbook, attaches a picture, and puts up a price and a location. We can assert if this is saved on the database.

(iv) Each want-to-buy post must include a title, a product description and a want-to-buy-price associated with the item. If this cannot be done, the test fails.

(v) After the request has been sent, the listing should show up in the database and be present on the website. If not, the test fails.

(vi) After posting, the posting should be shown under the “Want to buy” category and the general public can see that post. If this cannot be done, the test fails.

Uncommon cases:

(a) During the meetup, the buyer was not satisfied with the condition of the test book and wanted to cancel the transaction.

(i) The listing will be reposted on the website. We can assert if this is true.

(ii) The funds will be automatically refunded back to the buyer. We can assert the message returned by the paypal API.

2. **Student seller posting related acceptance testing**

The user input for student sellers will be a post of some kind of merchandise.

Common cases

(a) The seller wants to create a new post to sell one or a set of textbooks.

(i) The seller will go to “create a new post” tab of the website. If there is no such tab, the test fails.

(ii) The seller will enter a page to add required details about the textbook by clicking on the tab. If the page fails to load, the test fails.

(iii) The seller will be able to enter the name of the book, a text description about the book, at least one photo of the book, category of the book, and price of the book. If any of those fields is missing, the test fails.

(iv) The seller will be able to complete the post so that it is visible to every user. If the post cannot be seen after completion, the test fails.

(v) The seller should still be able to edit the post before it is purchased. If editing is not allowed, the test fails.

(b) After the textbook is sold, i.e. the seller has received payment from the buyer, the post should be automatically closed so that no buyers can see it anymore. If the post is still visible to any user, the test fails.

Uncommon cases

(a) The seller decides to stop selling one of the textbooks he posted before it is purchased.

(i) The seller will be able to edit his post. If editing is not allowed, the test fails.

(ii) The seller will find a button to remove the post so that no buyers can see it anymore. If any user can still see the post after removal, the test fails.

3. Student seller transaction related acceptance testing

The input is the update to the *product* database that a particular item for sale has been *in_transaction*. The buyer has already paid the cost of the textbook + fees through the Paypal API. That money is deposited in the website’s Paypal account temporarily before the transaction succeeds.

Common case:

1. After chatting with the buyer to coordinate a time to meet up, the buyer and the seller agree to set up a location and the time to meet up.
    a. If it is impossible for the buyer and the seller to agree to meet, then the CANCEL TRANSACTION process starts.
    b. Assert that TIMER process has started when the buyer has paid the money for the textbooks. If TIMER process runs out, then assert that CANCEL TRANSACTION process starts.
    c. Assert that CHAT process has started between the two parties

2. The buyer and the seller have met in person.
    a. If the buyer does not like the physical condition of the book or for other reasons, the buyer can cancel the transaction. Assert CANCEL TRANSACTION process starts.
    b. If the buyer does like the item, the seller can scan a QR code that was generated for this transaction (using OpenCV API). Assert that QR CODE VERIFICATION PROCESS is successful.
    c. Assert that the PAYMENT process to give money to the seller has started.
    d. Assert that the EMAIL process using SUCCESSFUL TRANSACTION template has been sent.

Uncommon case:

1. CANCEL TRANSACTION process is called.
    a. Assert that the REFUND process for the buyer has started.
    b. Assert that the EMAIL process using CANCELLED TRANSACTION template has been sent.
    c. Assert that the textbook in transaction is re-listed on the website.

4. Admin related acceptance testing

The user input is the period of which they want to see the revenue, which could be monthly or total aggregate. Depending on the input of the user, the system will show the number corresponding to the monthly, yearly, or total revenue of the website.

Common case

(a) User chooses *Monthly* revenue. It returns a view of monthly revenue for the past 12 months from the *Transaction* database.

Assert that: there are twelve returned values corresponding to monthly revenue.

Assert that months with lower revenue compared to previous month have color variable "Red."

Assert that the months with higher revenue compared to previous month have color variable "Blue."

Assert that screen title shows "Monthly Revenue."

(b) User chooses *Total* revenue. It returns a view of total aggregate revenue from the *transaction* database

Assert that there is only one positive value returned.

Assert that the screen title shows "Total Revenue."

Uncommon cases

(a) The admin does not have access to the revenue database. The system returns an error message.

In acceptance testing, we can suppose a case where the user does not have access to the database (it does not exist). Then we can assert that the error message code we received is the same as the error message code for not having access to the database (or database not found).

Part 4:
3
Take into consideration feedback from your IA mentor or other members of the teaching staff. If you disagree with the feedback, explain why. **Revise accordingly the list of specific facilities you plan to use for compiler/runtime (or equivalent),** an IDE or code editor, a build tool (or package manager if 'build' not applicable), a style checker, a unit testing tool, a coverage tracking tool, a bug finder (that's not just a style checker), and a persistent data store appropriate for your chosen language(s) and platform(s). If different members of the team plan to use different tools, please explain. It is ok to change your choice of tools later after you start developing your application.

Back-end:
IDE: Pycharm
Compiler: python 3
Package manager: pip
Unit Testing : unittest module
Style Checker: pep8
Coverage: Coverage.py module
DB: MySQL
Bug finder: DeepBugs (https://plugins.jetbrains.com/plugin/12218-deepbugs-for-python)

Front-end:
IDE: Visual Studio Code
Package manager: npm
Style checker: ESLint
Bug finder: ESLint
Language: javascript
Framework: Reactjs