This is a work in progress.

**Purpose**

To serve as a design doc for COMP 307 project.

**Business Problem**

Universities do not have online marketplaces for students to **easily and securely** buy, sell and exchange items.

**Use Cases (Requirements)**

1. As a student, I would like to buy, sell and exchange items within my university.
2. As a buyer, I would like to see a list of all items for sale.
3. As a buyer, I would like to add items to cart and checkout online.
4. As a buyer, I would like to see my purchase history.
5. As a student, I would like to chat with other students.
6. As a seller, I would like to add, modify or remove product listings, and see all of my listings.

**Features (Matching Requirements)**

1. Registration page. Email validation during registration using MailGun.
2. Product listing page showing all items for sale. Users can add items to cart.
3. Cart page showing total price. Users can checkout using PayPal or Stripe.
4. Purchase history page showing purchases.
5. Chat page for each user. Other users can join and leave messages. (In my opinion this is easier to implement for the frontend because we don’t need different tabs for chats with different users).
6. Control panel page that allows users to list products to sell (add, update and delete). All products belonging to the user are shown on this page.

**UI**

1. Login page
2. Registration Page
3. Product listing page
4. Cart page
5. Purchase history page
6. Chat page
7. Control panel page

**Backend Endpoints**

|  |  |  |
| --- | --- | --- |
| **API** | **Step** | **Comment** |
| Login  UI1 | 1. Validate user exists 2. Validate password |  |
| Register  F1 | 1. Valid university email 2. Check user does not exist 3. Proceed with MailGun validation | We can hard code email address like mail.mcgill.ca for now. This should be dynamically stored in database. |
| Get listings  F2 | 1. Select all from DB | If possible, items added to cart should be stored in browser cache. |
| Checkout  F3 | 1. Validate items are in stock. 2. Redirect to Paypal or Stripe. 3. Adjust items stock accordingly. | Frontend should handle total calculation and redirection to payment. |
| Get purchase history  F4 | 1. Select entries from db matching user id. |  |
| Get chat  F5 | ?? | I’m not sure if we need an endpoint for this. |
| Add/modify listing  F6 | 1. Put method | We can have input validation in the FE. |
| Delete listing  F6 | 1. Set item to inactive = true | Discussed that we should keep listing in DB so users can still view in purchase history. |

TO DO:

1. We might need a table for the cart, this would also affect how checkout works.
2. We should also allow users to remove items from cart.
3. We should also let sellers see their selling history. Maybe it’d be easier if we just called it transactions history instead.
4. This part needs to be revised, let’s take some time to see what makes sense from a user’s point of view.

**Storage**

There are a lot of trade-offs due to project requirements. We can only use relational DB.

|  |  |
| --- | --- |
| **USER** | |
| **Key** | **Value** |
| ID (PK) | Long, auto increment |
| Email | String |
| School | String (enum in backend) |
| *Address (?)* | *String (?)* |

|  |  |
| --- | --- |
| **ITEM** | |
| **Key** | **Value** |
| Id (PK) | Long, auto increment |
| Photo | ? (maybe this can be String of bits that we can encode back to picture) |
| Name | String |
| Price | Double (=String?) |
| Count | Long |
| Seller id | User\_id (relation) |
| Active | Boolean |
| Location | String (google map url?) |

|  |  |
| --- | --- |
| **TRANSATION** | |
| **Key** | **Value** |
| Id (PK) | Long, auto increment |
| Buyer\_id | User\_id (relation) |
| Seller\_id | User\_id (relation) |
| Item\_id | Item\_id(relation) |
| Date | Long (epoch millis) or Timestamp (?) |
| Count | Long |

**Stretch Goals**

1. Google Maps location service integration.
2. Product details page. Can be accessed from product listing, cart, purchase history pages.
3. User ratings.
4. Searching feature on product listing page.
5. Searching feature on purchase history page.

**Milestones**

1. Research phase
   * Date: February 13, 2020
   * Steven: setting up Django backend, account management, product picture
   * Trisha: Google Maps API, chat/websocket
   * Zach: UI, Cart feature cache