



Columbia Exchange Platform

Dongbing Han (dh3071)

Jiapeng Guo (jg4403)

Weiran Wang (ww2584)

Yuqi Zhang (yz3983)

Introduction




Every year, there are many Columbia students graduating and there will be a problem with their furnitures/ stuffs they do not want. We planned to build a platform that could provide Columbia students a reliable place to sell or buy second-hand stuff.

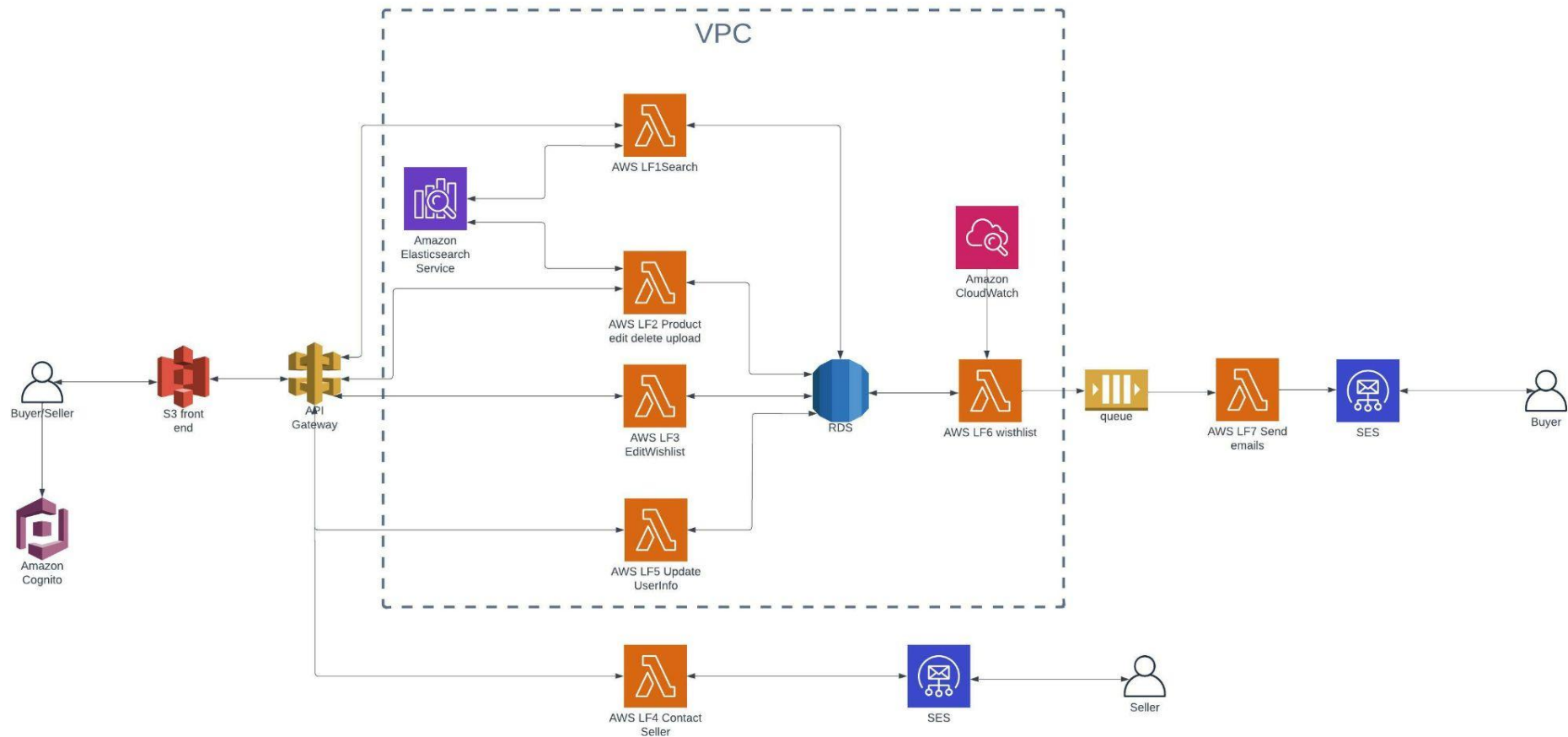
AWS Services

- ❖ We used **Cognito** to authorize the users and used **S3 bucket** to hold our website's frontend.
- ❖ We also used **API gateway** and Lambda functions to connect our frontend and backend and implemented our services.
- ❖ Our data is being stored in **RDS Database** and is being connected to our frontend by Lambda functions. All the **Lambda functions** are in the same **VPC** with RDS Database.
- ❖ We also use **SES** to send emails to users for their subscribed tags and also help buyers to contact sellers.
- ❖ In the following couple of slides, we will describe the functionality of our project as well as all important fronted and backed designs more comprehensively.

Supported Services

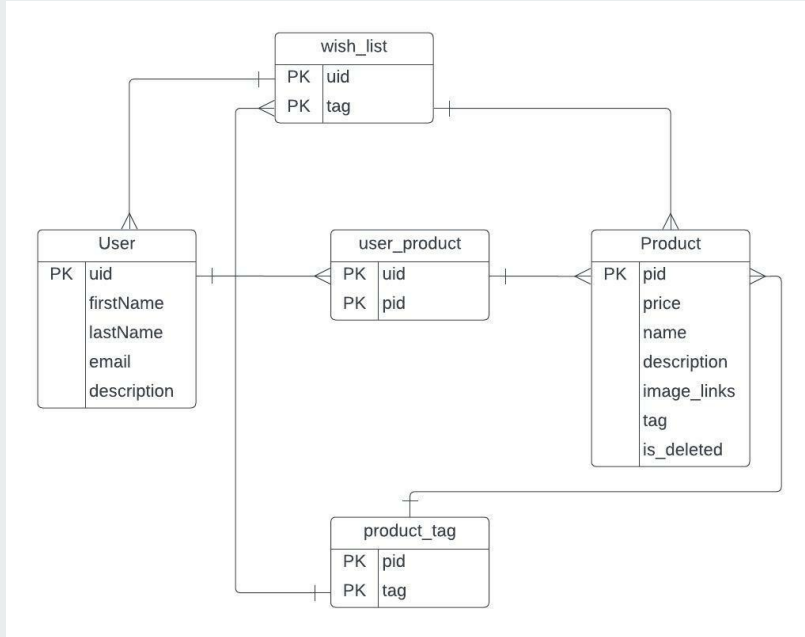
- 
- Our application is required to register with Lion emails, and once the user logs in, they are able to add, edit or delete products on their selling list through our platform and view the information in profile page.
 - Our application supports the user to tag the product when they upload the product so that others could find the product more easily. Other information such as name, price, description, and picture of such product is also support to upload.
 - Our application supports users to search what they want by typing into a search textbox. User can also click a item to view the detail of such product.
 - Our application supports leaving messages to sellers from buyers. Users can contact sellers on the platform and ask questions about items they want to buy. Seller will automatically received information that who is interested in his products and which product has a potential buyer.
 - Our application supports user to subscribe to tags that he/she is interested in and will receive emails about the information of the newest products of that tag that are added for sale in the last week.

Architecture of the project



Database Design

Following is our database design



- This is the design of our database. We have these 5 tables in a schema that is stored in a RDS database.
- The “**User**” table is gained from the Cognito User Pool. Everytime a new user registered, the user’s information will be stored in this table.
- Once a registered user uploaded a product, the product information would be stored in to the “**Product**” table. “**user_product**” table and “**product_tag**” table are two relational tables that would also be updated.
- When a user added any tag to their wishlist, the “**wish_list**” table will be updated.

API Design



- **/upload POST**

We use this endpoint to save user information in our backend database from Cognito

- **/productsearch GET**

We use this endpoint to do the search by queries

- **/upload GET, POST, PUT(DELETE)**

This endpoint is used for users to upload, edit or delete their selling products.

- **/wishlist GET, POST, DELETE**

We use this endpoint to get, add, or delete the tags in user's wishlist.

- **/getuseritem GET**


We use this endpoint to get all the products that are related to a specific user.


- **/contactseller POST**

We use this endpoint to send the message from buyers to sellers through email.

Lambda Functions

We have following important aws lambda functions

- 
- The **product-search** function takes in a query and use that query to search for products that either its tag or name can match the query, and return the detail of products to the frontend for showing.
 - The **upload** function can upload, edit, or delete products by taking in product details and update the backend database.
 - The **user_wishlist** function can add and delete user-tag pairs in the database. And it can also take in a query (uid) to search for all the tags that this user subscribed to.
 - The **contactSeller** function takes in all the information of buyer and seller and sends the buyer's message to the seller through SES.
 - The **updateUserinfo** function connects with register page and will store users information once they successfully register.
 - The **wishList** function is triggered by CloudWatch weekly. It will fetch all the tags that are subscribed by users and get all the newly added products. Then it will push the formed email and the contact information to SQS and waiting for the **wishlist_email** function to send the email.

- 
- **Index.html**, **confirm.html**, **signin.html** and **signup.html** consists the whole register, login and logout process which connect with aws cognito and RDS database to keep user profile information.
 - **profile.html** is used for user to view his/hers information after login. And also support user to upload a item or check items that he already upload.
 - **upload.html** and **edit.html** supports user to upload a new item to our exchange platform and edit/delete any of the item that he already upload.
 - **homepage.html** is the main page of our platform, and it shows the newly added products for the recent week.
 - **productDetail.html** provides more detailed information after user click any item. Information includes products name/price/description/tags, and sellers information such as their username and email address. There will also be related products showing on that page.
 - **otherProfile.html** provides user's profile for other users to view. Users can access other user's profile by clicking on the username showing in the **productDetail** page.
 - We also implemented a **Nav bar** on every page, and from there, users can access our search bar by clicking on “**Search**” button and access to our website information by clicking on “**About us**” button.



Thanks for Watching !