**Developer Documentation**

1. **Objectives:**

This assignment will expose you to the following Amazon technologies: Lambda, API Gateway, and DynamoDB

1. **Requirements:**

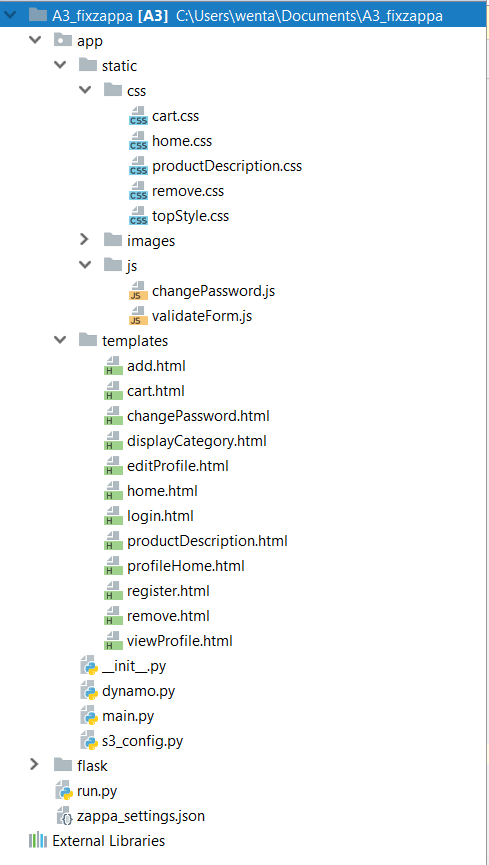
Use your creativity to develop a cool application that complies with the following requirements:

1. All application data should be persistently stored on DynamoDB and S3
2. The background logic should be implemented using Lambda functions
3. **General Architecture**
   1. **Browser:**

This web application is developed and tested using Mozilla Firefox web browser. The web browser in this architecture serves as the graphical user interface to general users and as the client to the Flask web development server. The main functionality of the web browser is to collect information from users in the format of forms and to display the transformed pictures as well as any additional information such as errors and warnings from the backend Flask web server to the users in a user-friendly environment. The HTTP post and put request methods will be issued by the web browsers to push collected information to the Flask web server and to retrieve information from it.

* 1. **Flask Web Server:**

The web server used for this web application is named Flask. Flask is a micro web framework written in Python which based on Werkzeug toolkit and Jinha2 template engine. The Flask web server in this architecture plays the backend web server role to process the HTTP request from the web browser and the client role to send requests to the DynamoDB database and the Amazon S3 for storing user and product data.



**Figure 1:** Project structure breakdown in Flask Web Server

* + 1. **Structure breakdown**
       1. **HTML and CSS**

All html files are stored under the A3/app/templates directory and the CSS stylesheet for each HTML page is stored under the A3/app/static/css directory. The CSS style sheet will be called when the corresponding html page is being rendered to the client.

The HTML and CSS sheet are programmed based on Bootstrap framework to take the advantage of its responsive grid system and extensive prebuilt components. This approach allows the application to readjust its components based on user’s device and therefore provides the best browsing experience across all platform.

* + - 1. **JavaScripts**

The JavaScript files stored under A3/app/static/js are used to perform changePassword and validateForm functions.

* + - 1. **Python files**

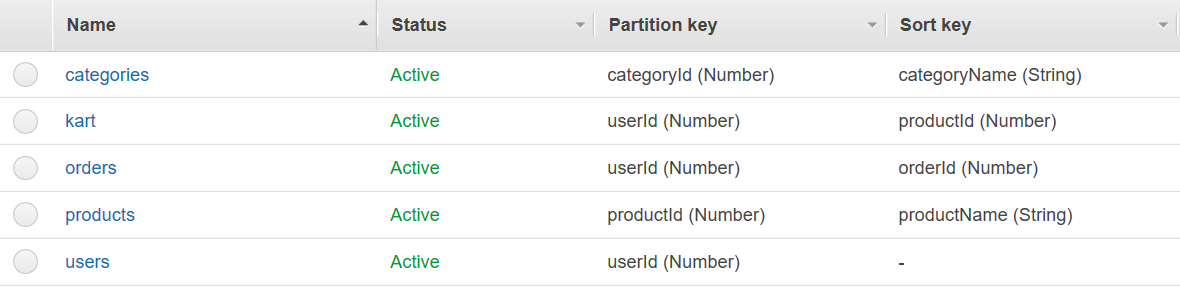
All python files which define the backend logic and to initiate the web server will be stored under the A3/app directory.

**3.2.2.2 Salt (Cryptography)**

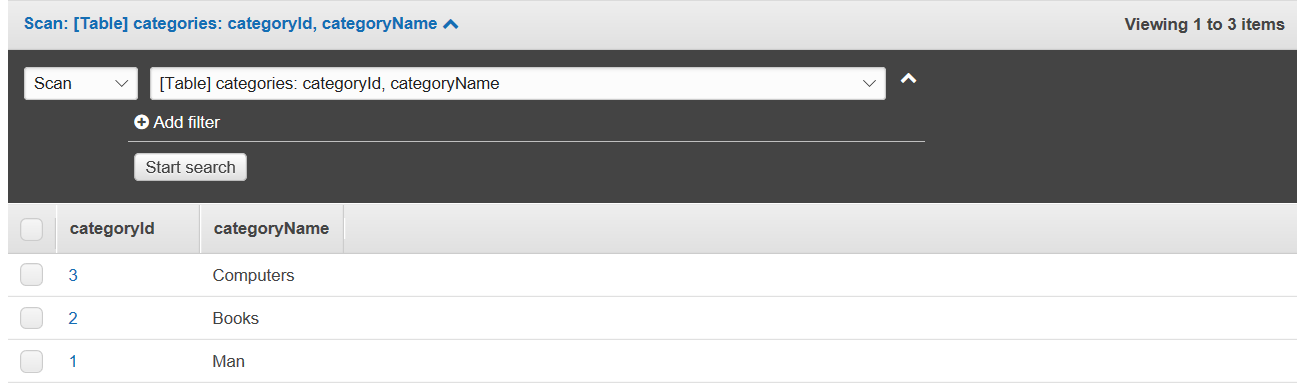
To store the hash of the password concatenated with a per-user salt value, hashlib module is imported to Flask in is\_valid, changePassword and register functions. The hashed password will be generated and saved to the backend DynamoDB database once a user finished creating his or her user account through signup form. When user enters his or her password through login page, a new hashed password will be generated by calling hashlib.sha512() function and this new generated password will be used to compare with the hashed password saved in the DynamoDB database. User can access his or her personal account only when the comparison result is true.

* 1. **DynamoDB**

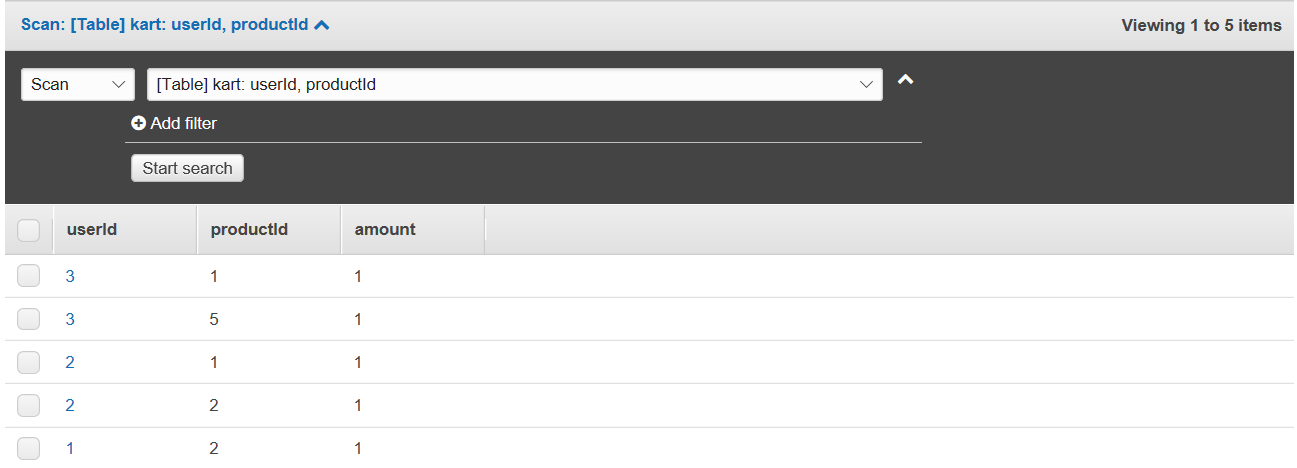
DynamoDB database in this architecture is used to store the information submitted by users during the signup phase, the information of products submitted by the administrator during the add item phase and the products each user added to their cart during the sign in session. Categories of each products is also stored in a separate table in the Dynamo DB to allow customers shopping items by category.



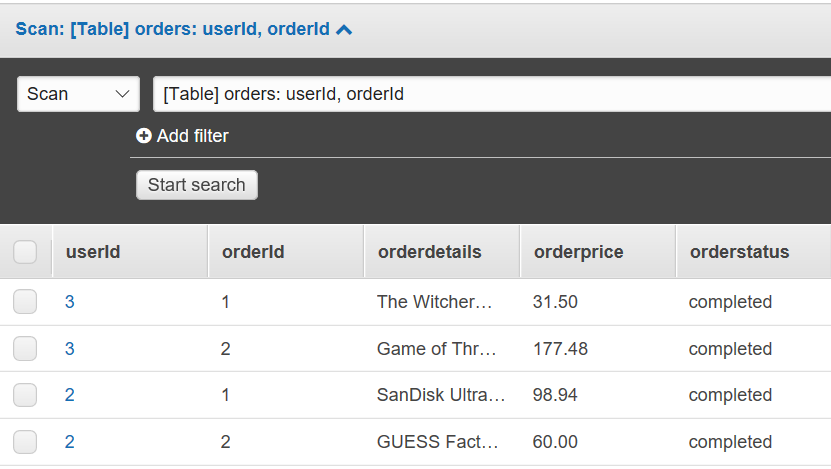
**Figure 2:** Tables defined in DynamoDB database



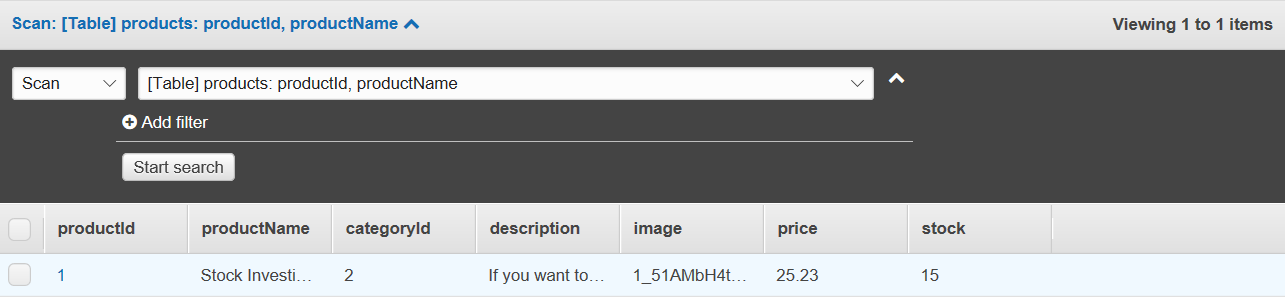
**Figure 3:** Schema of Categories table



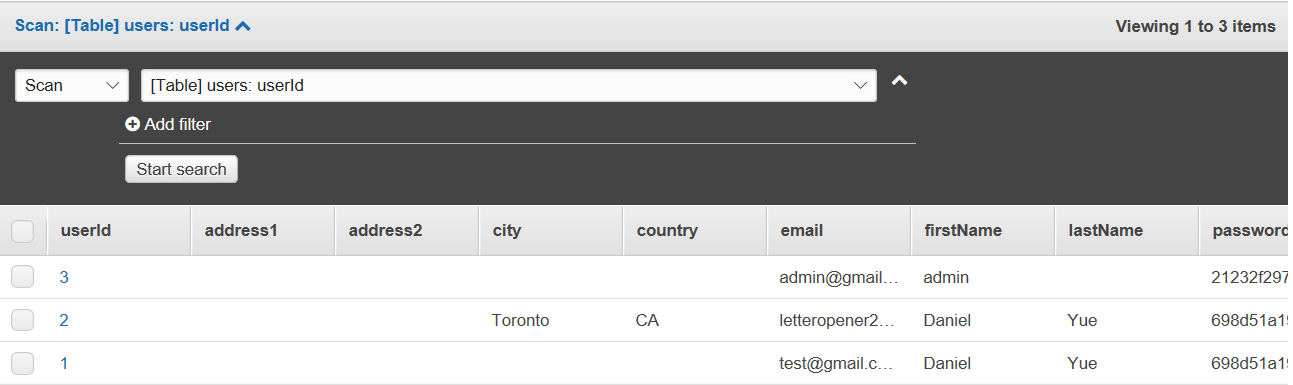
**Figure 4:** Schema of Kart table



**Figure 5:** Schema of orders table



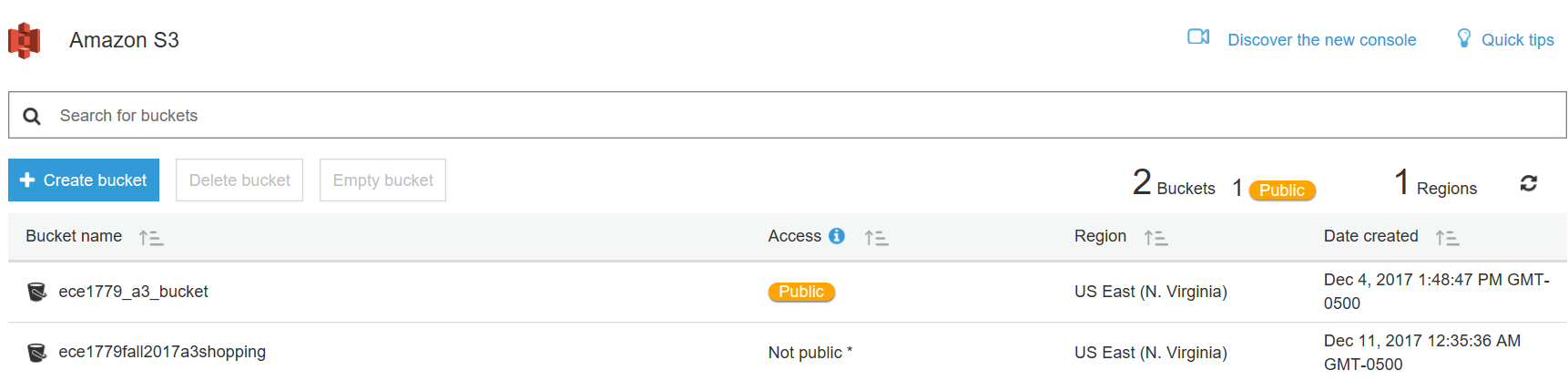
**Figure 6:** Schema of products table

****

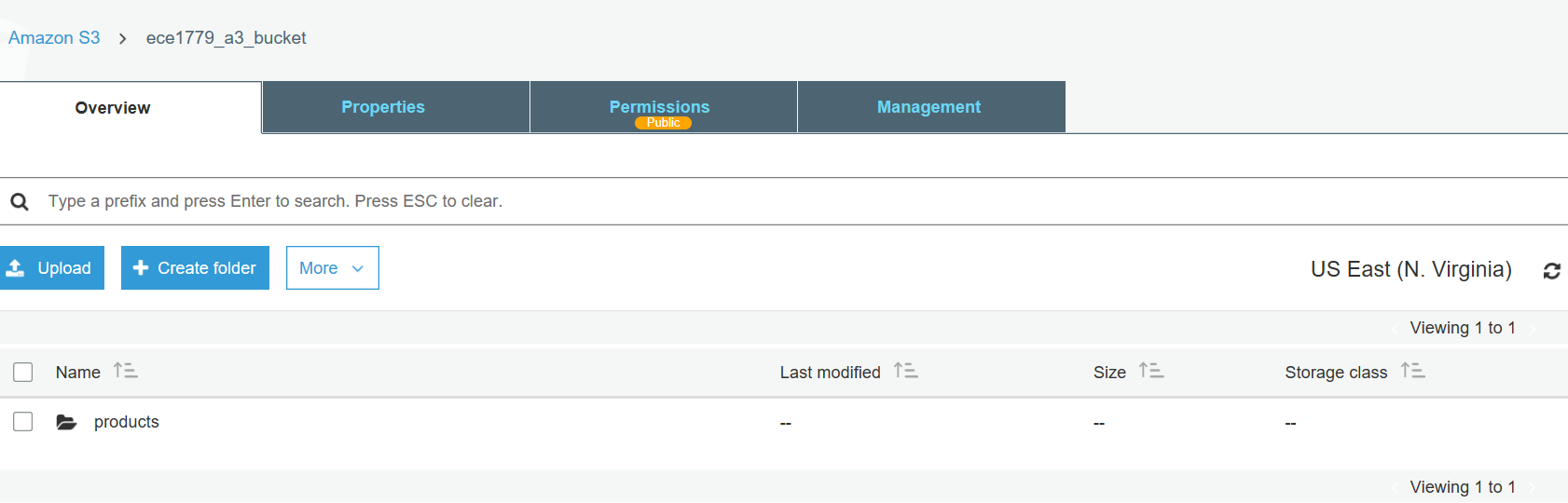
**Figure 7:** Schema of users table

* 1. **Amazon S3**

Amazon S3 in this architecture is used to store all the codes submitted by Zappa. In Amazon S3 console, there are two instances of S3. The **ece1779\_a3\_bucket** instance is used to store the pictures uploaded by the administrator during the add item phase. The **ece1779fall2017a3shopping** instance is the S3 bucket defined in the Zappa setting file and will be used as the archive of the flask code for Zappa deployment.

****

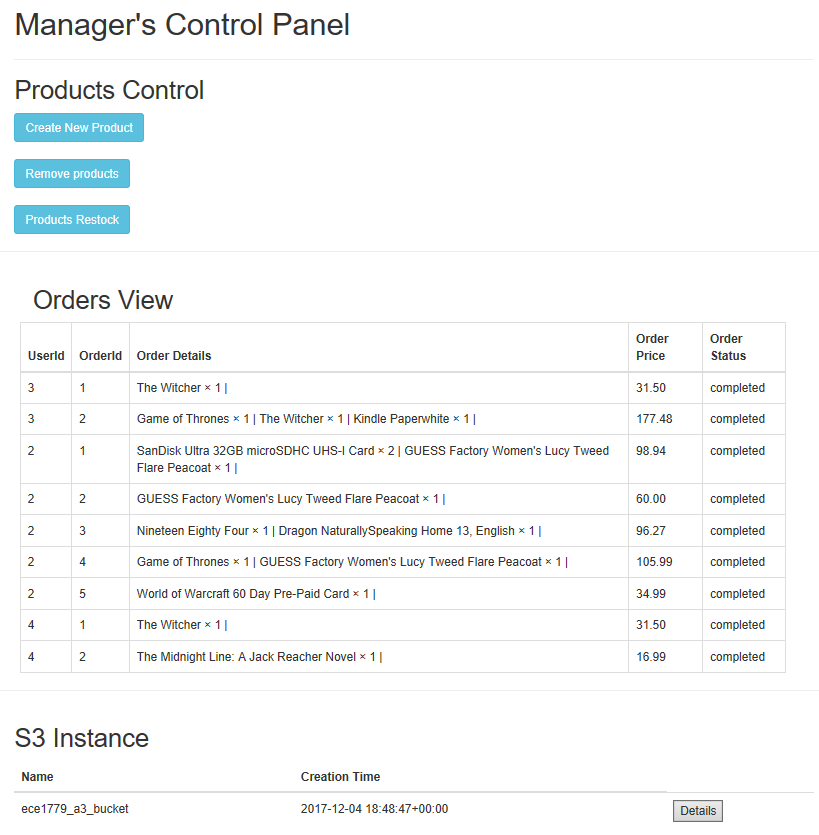
**Figure 8:** Amazon S3 bucket



**Figure 9:** Product directory created on Amazon S3 instance ece1779\_a3\_bucket

* 1. **Manager UI**

The manager UI in this architecture serves as the backend of the online shopping system application. Administrator could perform tasks such as create new products, remove products and products restock. Administrator also has the option to view the status of the S3 instance named **ece1779\_a3\_bucket** which is used to store the products’ pictures.

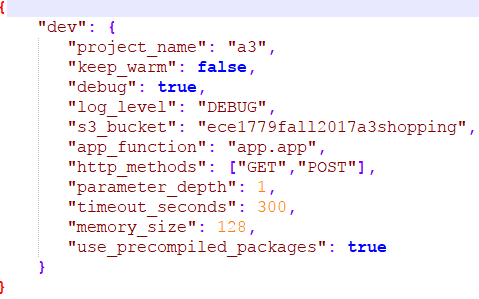


**Figure 10:** Manager UI interface

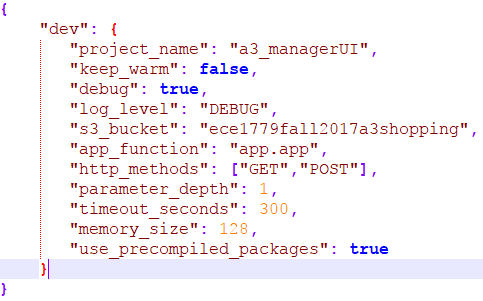
* 1. **Lambda/Zappa - Serverless Python**

Zappa has been used in the architecture to deploy Amazon AWS Lambda function. The flask application will be implemented as a lambda function on Amazon AWS and the function will be only triggered when there is a web request from browsers in the user side. The lambda function also provides scaling capability when the request traffic becomes high. The list below is the features provides by Zappa:

* Server with a 40-millisecond life cycle
* The server is online 24/7 and only process request when there is a request comes in.
* Each request is given its own virtual HTTP “server” by Amazon API Gateway
* Amazon AWS handles the horizontal scaling
* No requests ever time out
* Only pay for the milliseconds of server time that user use
* No need for load balancing or keep servers online ever again
* Free SSL certificates, global app deployment, API access management, automatic security policy generation, precompiled C-extension, auto keep-warms and oversized lambda packages.

****

**Figure 11:** Zappa Setting file of A3\_user directory



**Figure 12:** Zappa Setting file of A3\_manager directory