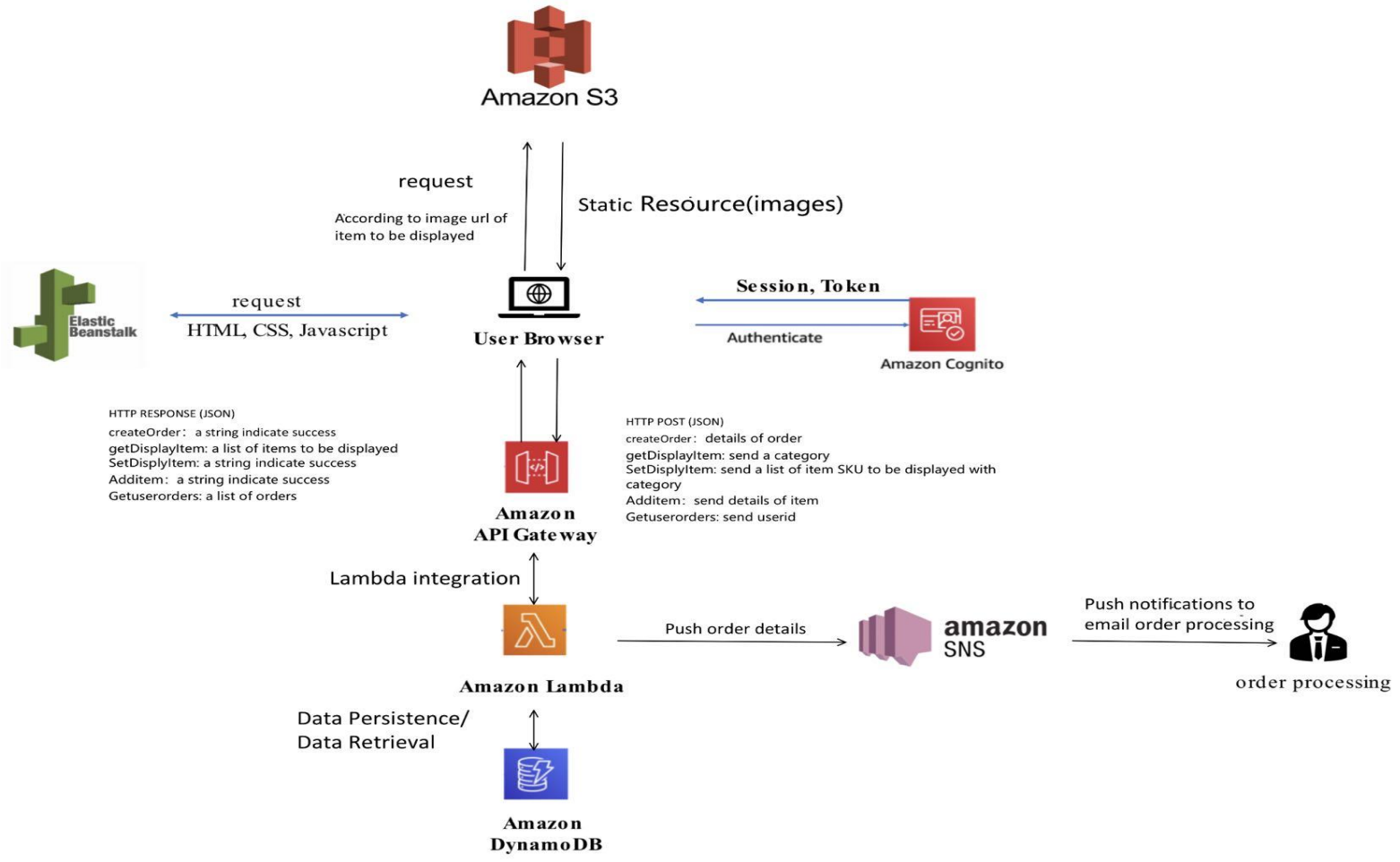
CSCI 4145 Cloud Computing Project
Report-Group 5

Group Member:

Archer Zhou B00806294
Junqiao Qu B00817232
Kessel Zhang B00809478

Final Architecture:



1. **AWS Lambda**: We deploy our backend code on AWS Lambda. We just need to upload our code and let Lambda manage the runtime environment. As described in the figure, lambda is integrated with API Gateway. When a request is sent to API Gateway, Lambda will be triggered to execute a predefined function. Also to support

different functions on the website, the function in Lambda will interact with SNS to push notification emails to all emails in a topic list. Besides, Lambda will read, and write the DynamoDB database to save and retrieve our item, and order information.

AWS S3: It will store the static resource of our website (images). The user browser will directly read images from the S3 URL. Currently, it does not interact with other AWS services.
AWS DynamoDB: It will save our relational text data in the database. It will only interact with Lambda to perform data read /write.
AWS SNS: It will push a notification when an order is created. It will only interact with Lambda and be triggered by the createOrder function.
AWS Cognito: It will manage our user information, like username, and password. And we use a Cognito application to manage our login and register process. Cognito will give a session when a user is successfully logged in. It will not interact with other AWS services.
AWS ElasticBeanstalk: We deploy our front-end code on it. It will send the user browser HTML, css, and javascript files. It will not interact with other AWS services.
AWS API Gateway: It is integrated with our Lambda function to build APIs. When a request is sent to API Gateway, it will automatically execute our back-end code in Lambda.

2. Data Storage: numeric, string data is stored in Dynamo
3.

Differences from the original proposal:

At the designing phase of our project, we plan to use the Secrete Manager to store our user login data (username and password). But after learning about cloud computing, we realized that the secret manager is not for storing the user data so that is not what we expected.

Many cloud services need a key pair or password to be accessed. A Secrets Manager is for creating and protecting a password for these application services. We may use this in our future work, but it is not the user login authentication function we need.

Instead, we decided to use AWS Cognito to manage our authorization module. It is very convenient, we could directly submit the user data to Cognito and Cognito will process the data for us. It could deal with both sign-in and sign-up functions, and we could also customize the requirements of user data based on our needs(such as limiting the length of user passwords).

Future Work:

1. As mentioned in the previous Security and Business Critical Analysis, the Cognito service we used follows the Transport Layer Security (TLS) mechanism, it secured our user information for login. But we have a potential risk which is our “creatOrder” API. We will calculate the total order price

locally and send it to the server which means people could interpret the request and change the order price.

We have thought of two methods to solve this problem, first is to encrypt the information, then it will be hard to figure out what the price is and also hard to change the price. The second method is to use lambda to calculate the order price instead of calculating locally. We can even combine these two methods, encrypt the information of the request and use lambda to validate the order price.

2. The other feature we want to add to our website is the real payment feature. We can implement the real payment function by calling the API of PayPal. We can use PayPal Sandbox to test and develop this payment feature.

3. We also hope to make future enhancements to our user identification information management, enabling users to change their passwords and change their email addresses. AWS Cognito has provided service for these features, we can use them to achieve enhanced user information management.

4. Last but not the least, we want to provide a feature to enable users to submit error messages and report order issues. We can create another function for this and use lambda to store the submitted error message to DynamoDB, the website owner can view these messages, solve problems and fix issues according to it.

Distributed Work:

Archer Zhou: Front-end Developer
Junqiao Qu: Back-end Developer
Kessel Zhang: Back-end Developer

References:

a