**School of Science and Technology**

**COSC2638 Cloud Computing**

## **Assignment 2 - Build a scalable app on Clouds**

Student: Ma Ngoc Phuong Anh – s3634707

Submission date: 17 May 2020

**A. Links:**

Project Web Link: <http://52.0.172.177:3000/>

Project API Endpoint Link: <https://uc2acbxet8.execute-api.us-east-1.amazonaws.com/dev/api/>

Project source code: <https://github.com/alicerose-ma/shopApp>

**Customer Credentials:**

Email: [guess1@gmail.com](mailto:guess1@gmail.com)

Password: 123456

**Admin Credentials:**

Email: [admin1@gmail.com](mailto:admin1@gmail.com)

Password: 123456

**B. Summary:**

In the era of online business, having a website to sell goods is some of normal things that every owner can archive in just some simple click with really cheap prize. However, to join in E-commerce world, an online store must be more advance than the traditional one. My application is to build an online flower store with strong website backend. That can handle a “tsunami” accesses at one time with reliable. Using Micro-service architect and exploiting AWS services (Lambda, DynamoDB, EC2,etc), I can easily done the work and ready to commercialize it.  
**C. Introduction  
*i. What are the motivations behind your idea?***

In the latest e-commerce market report in Vietnam (of World Economic Forum), the market is expected growing up to 10 billion dollar in the next 5 years. With that good sight, I believe it’s a huge chance for my family business, a flower shop, to deliver fresh buds to the far side of our country. Therefore, enabling the shop to online world with the help of new technology is my very first step.  
***ii. What it does?***

My online flower shop focuses on attractive design, fast response, which promote the customer’s pleasure feeling about just-picked flower sold in just updated website, but controlled by simple tool. I can post new product with just one click and help buyer proceed the cart quickly.  
***iii. Why it is required?***

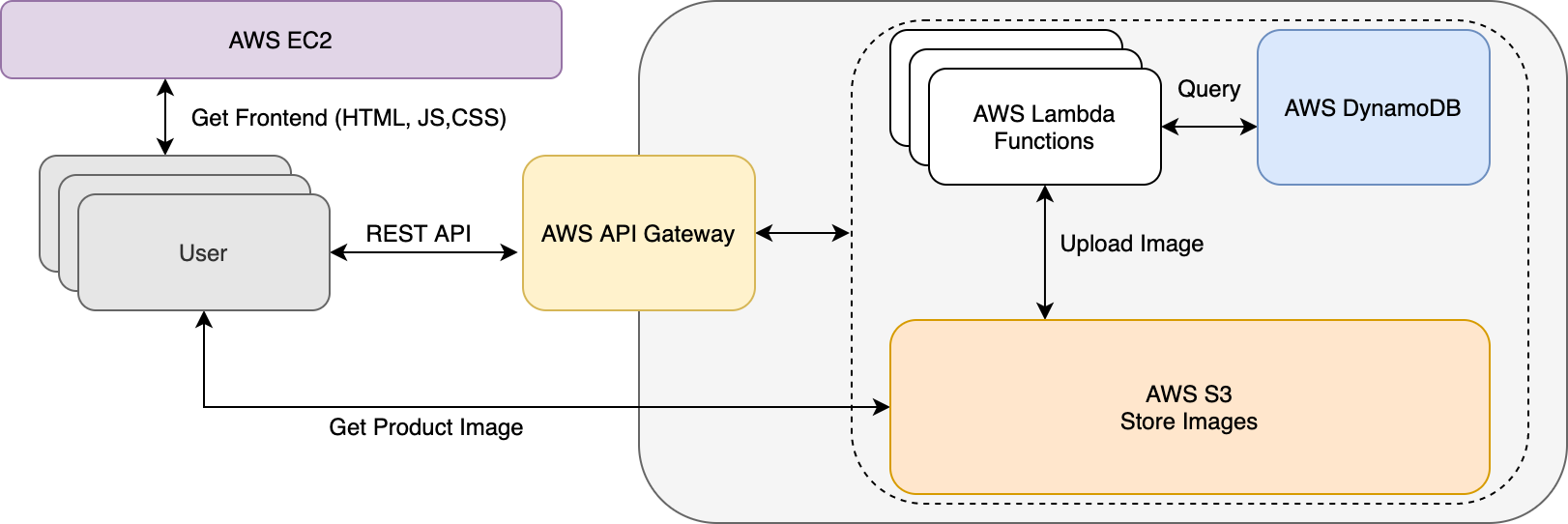
From experiences in online shopping, the fresher product is, the quicker we want to have it. Especially flower, people accept to pay more for faster ship or fresh-cut ones. However, I want my customer have better sense for my shop as its product is cheaper with deal but still fast-delivered and good looking buds. For example, in the mother’s day, no one expects their sweetheart mother to be wait too long for a rose bud, just because there are so many others try to get one from the same favorite shop and sometime cause the systems paralyzed.  
***iv. How it can be used as real-life application?***

All the flower shop owner should have thought about this feature for their business but not many have enough resources to promote it. Thus, my shop can be standard for who’s eager to expand their business.   
***v. The advantages/positive/new things of your application.***

Focusing on quick response for customers despite the numerous request at a time, my application benefits from new technology, the micro-service architecture is fundamental on Serverless framework. And I choose AWS Lambda service to run this new thing.  
***E. Related work: Refer some related works similar to your application.***

It’s easy to catch the similar solution on TIKI in their crazy-like event, called “Giut Co Hon”. Million people access the platform at same time for best deal. Without micro-service, it seems really hard to handle that circumstance with small system.  
***F. Software Design/Architecture***

***i. A high level architectural diagram that shows the communication between different cloud components used in your project and purpose of using those components.***

******

Backend architecture with AWS Cloud Services

**AWS API Gateway**

AWS API Gateway is used to create RESTful APIs as well as manage the schema of HTTP API and connect endpoints to respective backend services. It is able to trigger a serverless function directly through HTTP request, so it enables the truly serverless architecture for web application. Moreover, the AWS API Gateway can integrate with many other AWS services like AWS Lambda, AWS S3, DynamoDB.

**Microservices architecture – Serverless framework - AWS Lambda**

In order to improve the performance of the application when there are a large number of requests from the users at the same time, Micro-service architecture is used to break the entire functionality of the application into independently deployable services. As a result, each service performs particular functions and has its own logic and the database.

Therefore AWS Lambda is chose to use to allow the code can be run without dealing with servers in cloud (Serverless). An event triggers a Lambda function and dies after execution.

**AWS EC2**

Virtual machine run docker image of nodejs web application (frontend).

**AWS S3**

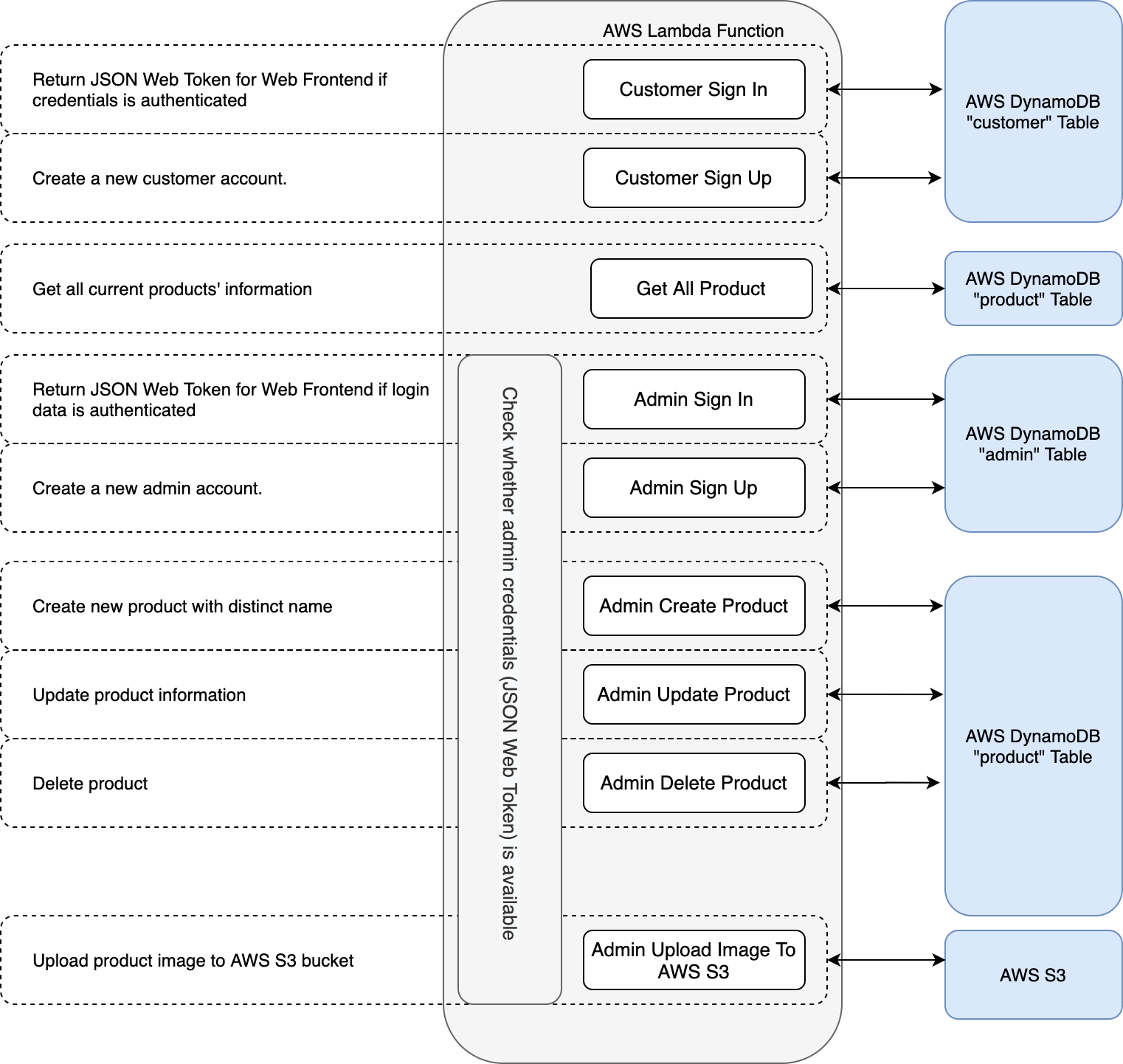
Store product images

***ii. Description of your dataset/data structure/APIs/sensors you used for your project (if any) [use figure if required]***

Application uses AWS DynamoDB to store data, containing 3 table: “product”, “customer”, “admin”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| "product" table | | "customer" table | | "admin" table | |
| Field | Describe | Field | Describe | Field | Describe |
| productId | UUID, used as hash key | customerId | UUID, used as hash key | adminId | UUID, used as hash key |
| name | Name of product | address | Address of customer | email | Email address of admin |
| des | Description of product | email | Email address of customer | password | Encrypted with “bcrypt” |
| image\_url | Product image link, stored in AWS S3 | password | Encrypted with “bcrypt” | createAt | Created timestamp |
| sale | Discount percentage (%) | phone | Phone number of customer |  | |
| price | Price of product | createAt | Created timestamp |
| createAt | Created timestamp |  | |

**REST API:**



Backend REST API with API Gateway, DynamoDB, S3 and Lambda

**G. Implementation**

- **Developer Manual**: A step-by-step guideline to reproduce your project [use figure if required] and make it live. This is like our tutorial sheet. For known/general description (e.g. creating and MySQL RDS instance in AWS, deploying project in Elastic Beanstalk, deploying project in Google cloud) you can refer to any web link directly. You can also refer to tutorial sheet if you have similar steps in your description (e.g. deploy application in Google cloud).

From step 1 to step 3, building website backend should follow the tutorial at this link: <https://serverless-stack.com/#table-of-contents>

**Step 1: Setting up the serverless backend**

* Create a DynamoDB Table
* Create a AWS S3 bucket
* Set up Serverless framework in Nodejs

**Step 2: Building Serverless Rest API**

* Create customer API
* Create admin API
* Check whether admin credentials (JSON Web Token) is available.
* Adding *iamRoleStatemements* to *serverless.yml*
* Handle API Gateway CORS errors

**Step 3: Deploy the backend**

* Deploy API
* Test API with Postman

**Step 4: Building express app (Nodejs)**

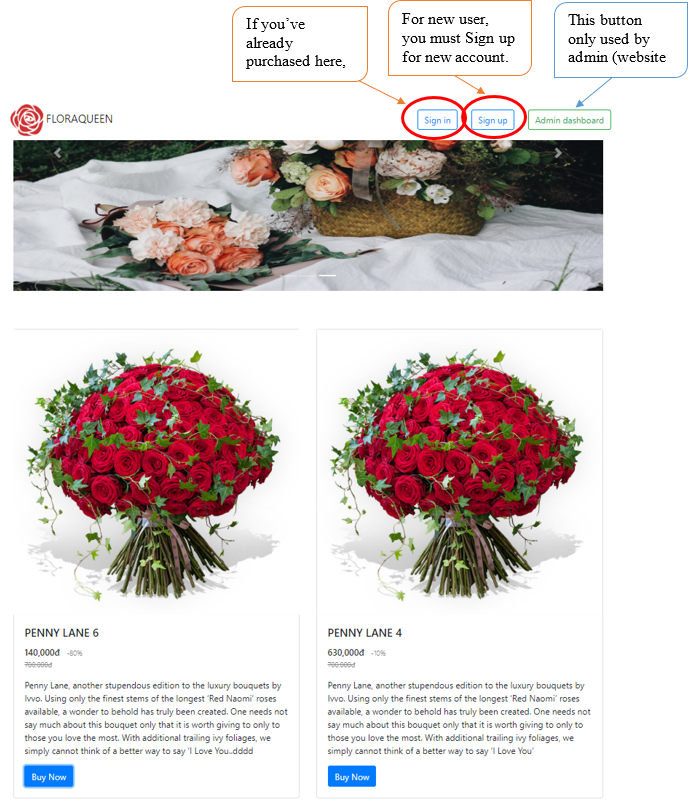
* Set up a express app (Nodejs) with ejs template, bootstrap 4 and jquery
* Communicate with API Gateway through “fetch”
* Create Admin UI
* Set up secure pages: redirect on login
* Create Customer UI

**Step 5: Deploying express app (Nodejs) on EC2 base on document at below link:** <https://nodejs.org/fr/docs/guides/nodejs-docker-webapp/>

* Create EC2 Instance
* Dockerizing a Nodejs web app
* Setting up EC2 to run docker image
* Create and associate Elastic IP address

**H. A small user manual:** **A quick overview of how to use your application.**

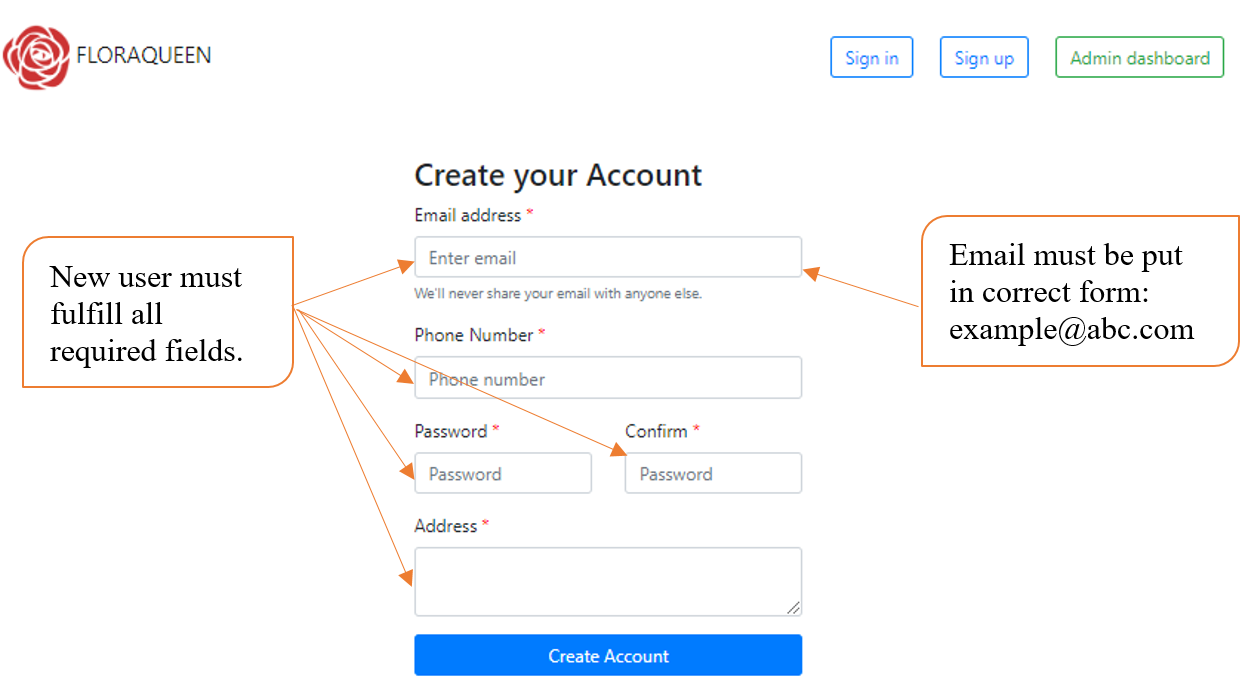
In home page:



* **As customer role**:

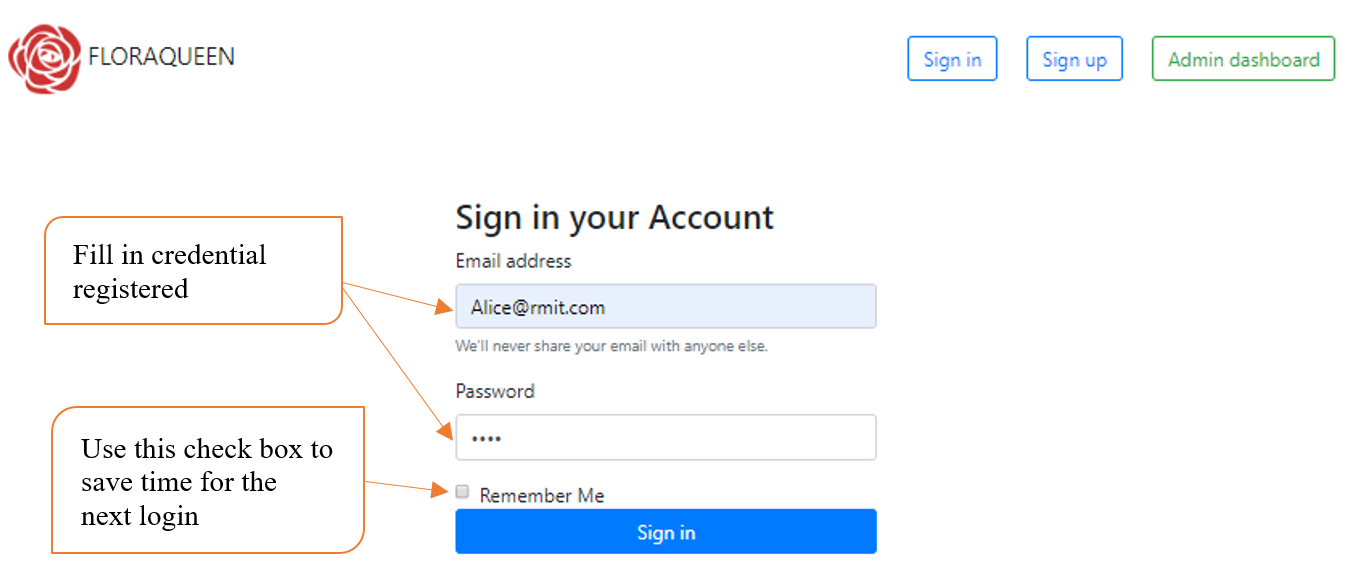
Normal customers can freely view the website with bunch of products displaying with attractive description, price and irresistible discount rate. If you want to buy one of these, you must sign in to add product to cart

* + For new user, after click Sign Up, new window will show up like below:



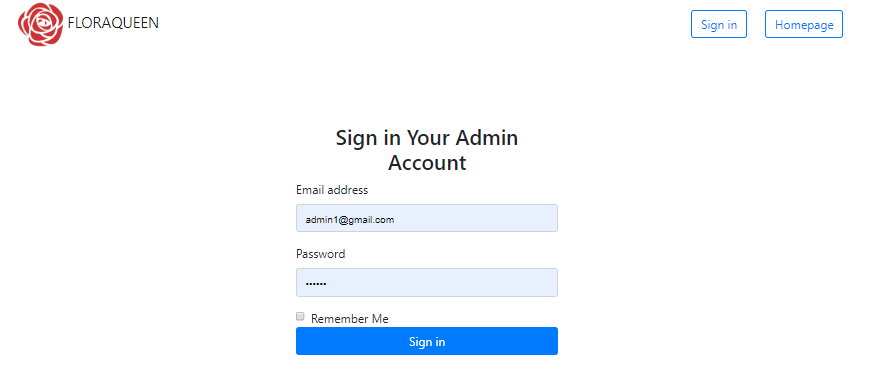
|  |  |
| --- | --- |
| In successful signing up, the website will return like below: | In fail registering, the website will return “Params Error”: |

* + After signing up, users should click “Sign In” button to get join in my flower world:



* **As admin role**:

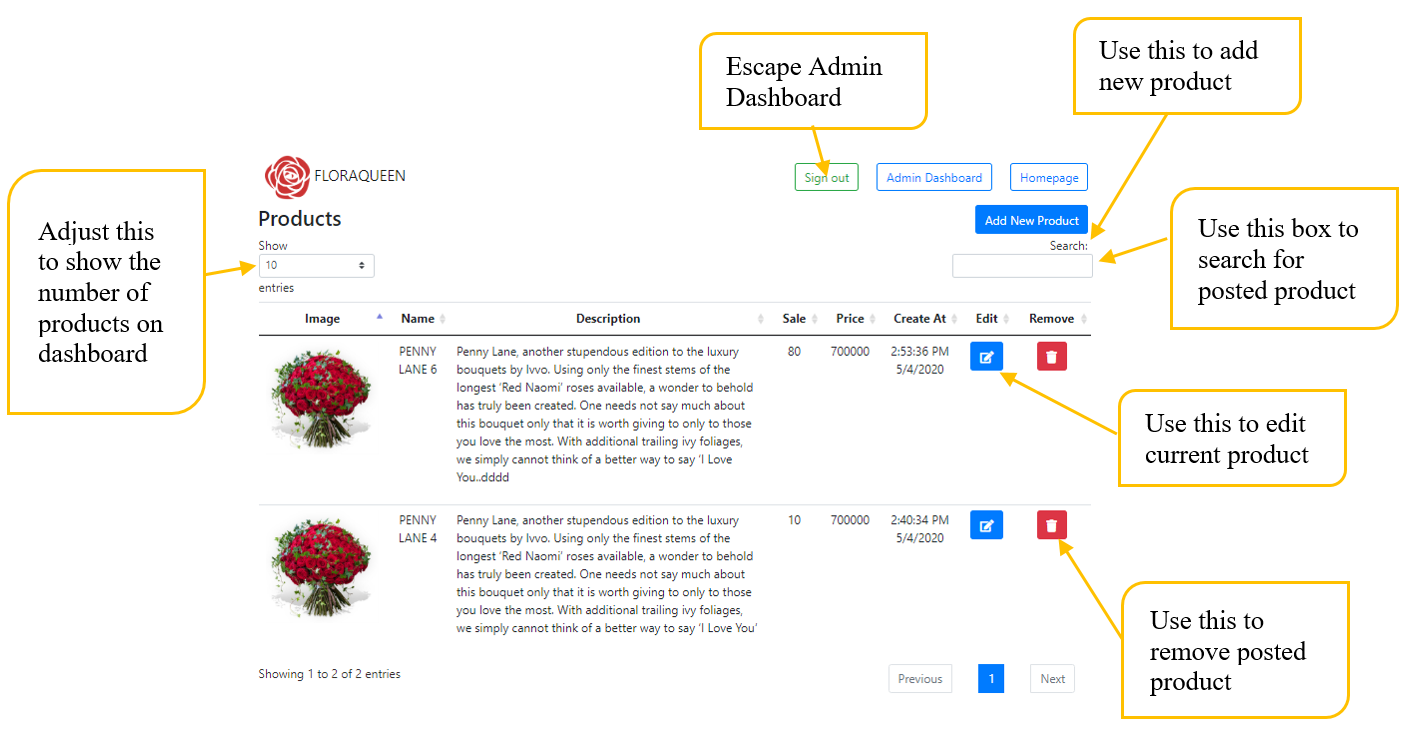
The owner use “Admin dashboard” to manage the business, adding, removing and modifying the product information. But first, owner must login:



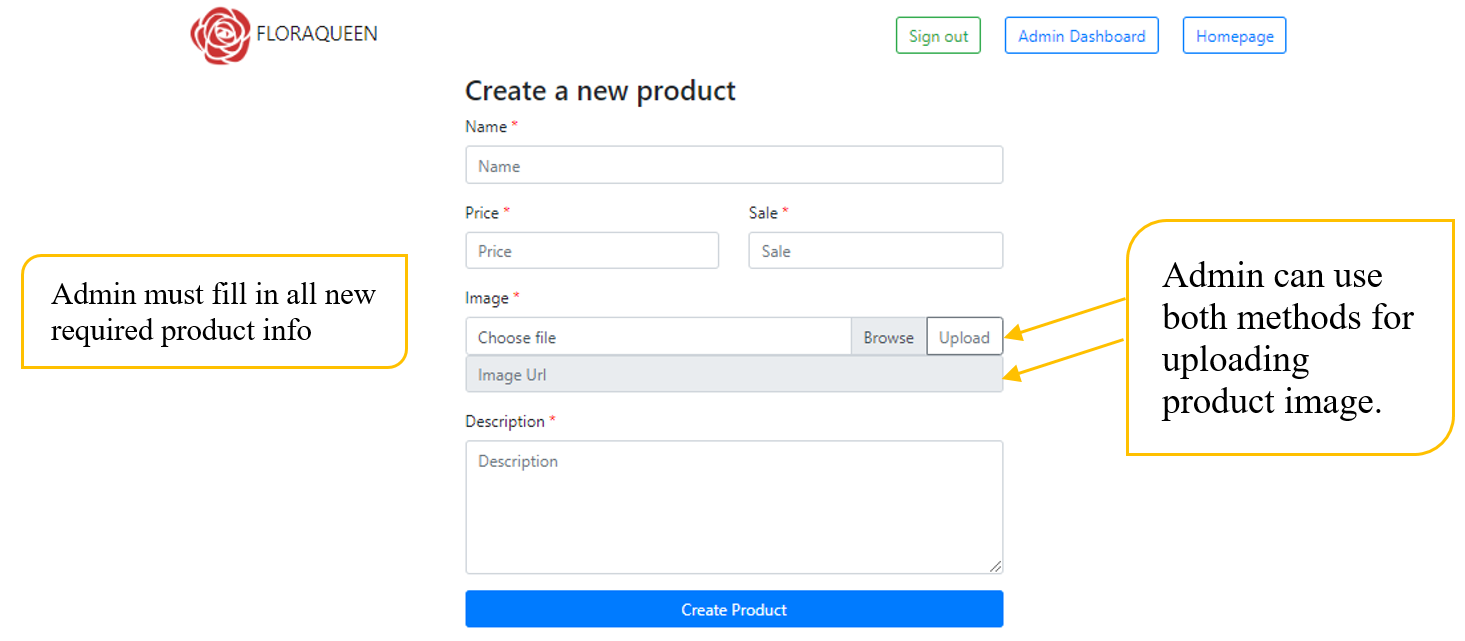
Fill in admin credential registered

Use this check box to save time for the next login

After successfully login, owner will see the managing site:

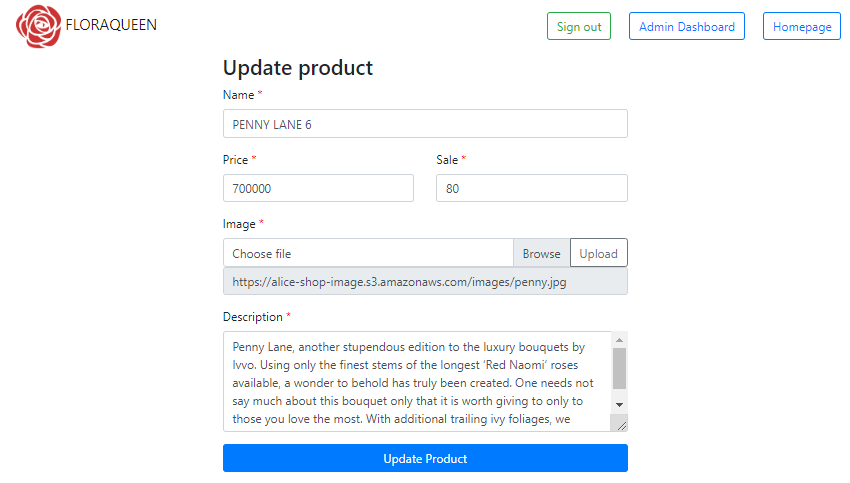


* + For adding new product, owner click on “Add New Product” button, the you will be redirected to this site:



Then, hit the “Create Product” button at bottom, the new products will appear in Admin Dashboard will full information.

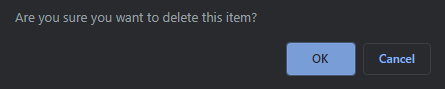
* + For editing current product, Admin clicks on blue button  to access this site:



Finish editing with this button

Admin edits current product infor, should not let any field empty

* + For removing product, just click the red button . A pops up appear to confirm:



Click “OK” if this is not a mistake.

**I. References: Important references/website links that you use to develop your application.**

References:

<https://serverless-stack.com/>

<https://www.serverless.com/blog/node-rest-api-with-serverless-lambda-and-dynamodb/>

<https://www.serverless.com/framework/docs/providers/aws/guide/functions/>

<https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/using-promises.html>

<https://aws.amazon.com/sdk-for-node-js/>

<https://microservices.io/patterns/microservices.html>

<https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html>

<https://medium.com/@mayneweb/upload-a-base64-image-data-from-nodejs-to-aws-s3-bucket-6c1bd945420f>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithDynamo.html>

<https://github.com/auth0/node-jsonwebtoken><https://github.blog/2020-04-15-npm-has-joined-github/>

<https://stackify.com/aws-lambda-with-node-js-a-complete-getting-started-guide/>

<https://www.serverless.com/amazon-api-gateway/>