***Readme***

***Acknowledgement:***

Thank you for giving me an opportunity to take the developer test.

***Problem Statement :***

Here at Bee, we are content heavy, so this project is designed to test how well you can manage content. Here, the front-end would like to send a JSON encoded API request to your server in order to upload a photo to the site. It is on you to create an endpoint that can handle this upload, then find a way to view all the photos that have been uploaded to the site in an efficient manner.   
  
Requirements:   
-Django or Node.js Framework   
-Photo upload endpoint   
-View photo endpoint   
-View photo feed endpoint   
-Authentication system that only allows authenticated users to upload photos but all users to view photos   
-Automated testing   
-Code coverage does not need to be complete, just generate that some critical functions of the system have tests attached to them.   
-Write-up on the systems design pattern choices   
  
This should explain framework decisions, where the system works scales well, and where the system could improve. This should be a quick project, so it is expected to be functional but not flawless.

***Solution:***

Assuming that the API request from JSON is sent and that the JSON is sending the image size to be uploaded along with the json to the server I can see the memory size and then decide depending on whether the image can be allowed to be uploaded to server. The communication with client would be through an ACK Code which is predefined.

I am sorry I was unable to create this scenario in the test.

***Design and Development of the Solutions:***

Used the following :-

1. Django framework.
2. Sqlite3 database.
3. Plain HTML and very little javascript.

Django Framework : It was quick to deploy a web application in a short time.

Database : A small database was required to store the admin credentials and only allow admin to upload images to server.

The Database has two tables which are user tables and image table. The user table stores the user name and password and level of permission as to who can be allowed to upload images to server.

The photo table is mainly used to store the image url and render the image respectively.

***Logic***

I have developed a page name home.html which is universal to all and any user who gets access to the URL can load the page. This page mainly contains image data and the images get dynamically loaded on the web UI.

I have created another page called login.html. Only authentic users get access to login.html. This page has the login page where user is asked to enter his username and password. If he is the right user with right permission, then he will see home.html wherein he gets to see the upload functionality which was disabled for others. In this way only authentic users can have admin rights to upload images.

I have used an input text which is hidden to store the permission value and use it when required. If a permission level is 1 then it means that the user is the admin and he has rights to upload images to server.

***Scalability***

1. If there are multiple servers then Nginx can be used and rules can be defined for nginx and based on the rules the requests get redirected automatically to respective servers.
2. A concept called lazy loading can be implemented, assume you have 100 images to be shown to the user. It is not necessary to load all images 100 images at a time but rather images can be loaded in chunks and as and when a user scrolls down the page, the next set of images get loaded. This enables faster loading of pages on the web UI. This is very suitable when the number of images are very huge.

***How to use the application :***

1. Unzip the mailed zip file
2. Open the folder named “website”
3. Run the runserver.bat file
4. CMD promt will open which will tell you the url as to where you can access the web application.
5. By default, it takes you to the home page which shows the list of images.
6. For admin rights, user has to use the following url

Application\_url/login , this will open the login page.

1. User is asked to give his login and password,by default I have given username = admin and password = admin. This will allow a user to login and after successful login, user can see the upload button along with the list of images.

Example : <http://127.0.0.1:8000/photos/> ( Opens the general page)  
<http://127.0.0.1:8000/photos/login/> ( Opens the login page)  
username: admin ; password : admin

***Areas for improvement***

1. Django is tightly coupled and does not have much flexibility, maybe a better framework.
2. User authentication developed is not safe, get methods should be replaced with post method and the URLs should be encoded.
3. User details are not stored in session due to lack of time, that will actually enable a developer to easily develop many features with one key value, permission.
4. The UI is not good.

Overall if given more time I could have developed a better product.

Waiting for a response from you

Regards,

Sumedh Datar

Mobile : 6824084071