Small Garden Project

Hi, my name is Donatus Omondi F17/1798/2019. I did the following project outlined below for my BSc. EEE Second Year Second Semester Computer Science Unit.

Abstract

Achievements

- An entity relationship diagram with all necessary constraints
- A database-backed web application enabling CRUD operations
- The queries that need to be turned in have been well defined and applied

Features incorporated in the web application

- Ability to create a user account
- Ability to log in to your user account
- Ability to log out of your user account
- Ability to add or update employee and customer information to the user account created
- A products and product details page for each product
- Ability to update or delete records in ALL the defined tables
- Login required feature when viewing any table page

Features omitted in the web application

- Ability to add records to all tables apart from the user, customers and employees tables due to constraints of time to create error handling on each table in case inappropriate values want to be added. This consequently affected the ability to do the following:
    - Make reorders on the web
    - Place phone orders on the web
    - Place any order in general
    - Add products to cart
- Ability to add customer payment info, such as debit or credit card or M-Pesa. This just happened due to constraints of time.
- Ability to view cart items
- Ability to make purchases
- An about page. Unfortunately

Introduction

Currently having happened to be working in a farm, I have encountered a number of challenges. My project, titled Small Garden, was made upon consideration of the following factors.

- The need to track orders and order items
- The need to track payments received with respect to a certain order.
    - It may happen payments are made partially, so all payments regarding a specific order to be noted
- The need to track in business transactions. These include :
    - Disbursing funds to employees
    - Expenses incurred by the business
    - Transferring anything from one point to another within the business such as funds from M-Pesa to a bank account
- The need to have all the information regarding the business available to partrons/investors. This includes:
    - Challenges being faced at the moment
    - Actions being taken regarding the challenges if any
    - Monthly reports about the progress of the business
    - Goals being worked towards
- The need to counter the tedious physical file system in general
- The need to reach more people. This consequently enables:
    - Reaching a larger customer base
    - Reaching a wider information base about better farming ideas
    - And many more benefits regarding the same
- The need to take advantage of setting an online footprint in general as one advantage has been highlighted above.

Related Work

A number of related works I have encountered fall under being only a market place for farmers to sell their produce or being an informative source for better farming ideas implemented by farmers.That of which makes sense since not a lot of farmers are tech savvy per say. Websites like Mkulima Young, The Poultry Site implement the topics as have just been said. However, kenchic deals with both tending to farmers needs as well as non farmers needs which is what I also had in mind. They just happened to focus on poultry. They as well have not implemented a way to purchase products they deal in as of this time. They only have a contact set up through which you can place orders.

Having said that I’m not sure on the tools the above businesses have used to achieve all they have achieved with regards to setting up their webpages.

Methods

Concerning the making of this project I applied the following:

- Programming the database-backed web application:
    - Python
    - Flask
    - CSS, HTML and Javascript
    - Docker and Docker Compose
    - Two kinds of RDBMS, Sqlite and Postgres
- Hosting the application
    - Linode
- Forming the entity relationship diagram
    - Microsoft Visio

Directory Structure

Enterprise

- db
- proxy
    - conf.txt
    - Dockerfile
- web
    - the_enterprise(python package)
        - img
        - static
        - templates
        - (from here now there were necessary python files)
    - Dockerfile
    - Requirements.txt
    - run
- docker-comopose.yml

Regarding Docker

I chose to use Docker and Docker-Compose to containerize bits of my whole application and link them together. Doing that enabled me to host them easily on a server, which I did using Linode. Docker configurations are defined in the Dockerfile file and docker-compose.yml file.

Regarding the Database 1.0

I chose to use two kinds of RDMS, Sqlite and Postgres. I used Sqlite when running the application on debug mode and Postgres when running the application when debug mode is off, i.e, on production. This enabled me to not interfere with the database that is active on production whenever I'm performing maintenance to the application until all the added/removed features seem to have worked out okay to move into the production application.

The Postgres database is configured on runtime and is stored in the db directory.

Regarding Proxy

The proxy directory is where all the ports regarding the application are handled. Web servicing to say it better. The database and the flask application each have a port but the only port that should be made public is one for the software that controls all the other ports. The configuration for this was defined in the conf file

Regarding the Python Package

This is where all the code went. All of this was now accessed through the run.py file that is outside this directory

Regarding the Database 1.1

I had a total of 30 tables made regarding the project. Since the SQL statements are rather long I have them on the zip folder containing the project code that is to be submitted. The tables are described as below

- User – Contains all users registered on the website
- Customers – Contains customer information of a user
- Customer Payment Info – Contains information on all the payment methods the customer has provided
- Office Designation – Contains the classification names that a given office should have at least one eg store, warehouse, etc
- Office – Contains all the offices the enterprise owns, be it a store, warehouse, etc
- Employee Designation – Contains the classification names that a given employee should at least only one eg, manager, customer service, etc
- Employees – Contains employee information of a user
- Other Business Designation - Contains the classification names that a given external business that interacts with us may have eg shipper, cleaner, repair, security
- Other Business – Contains information on all businesses that either interact with us or are of our interest eg Easy Coach, Jeff Hamilton etc

- Product Package – Contains the package names that a given product may have eg Picnic Special, Easter Special etc
- Product Types - Contains the classification names that a given product should have eg Equipment
- Products – Contains the information of every product
- Inventory – Contains the inventory information of every product
- Reorder Status – Contains the classification names that a given reorder should have only one eg Filled, Pending etc
- Reorders – Contains the information of every reorder made by the business
- Order Status – Contains the classification names that a given order should have only one eg Filled, Pending, Cancelled , Aborted etc
- Orders – Contains the information of every order made by customers
- Order Items – Contains the information of every item under a specific order
- Order Shipment – Contains the information of every order that is under shipment
- Invoices – Contains the information of all invoices sent to customers
- Payment Methods – Contains the payment method names that a given payment should have only one of eg M-Pesa, Debit Card etc
- Payments – Contains the information on the payments made by customers regarding a specific invoice
- Direct Purchases – Contains the information of every purchase that has been made by a customer in any office, be it a store, warehouse, etc. Direct purchases need no shipment, or invoicing and are paid there and then
- Direct Purchase Items – Contains the information of all the products under a given direct purchase
- Product Images – Contains the information on all images under each product
- Image Designaion – Contains the classification names that a given image should have eg home_page_image, register_page_image etc
- General Image - Contains the information on all images under use in the website
- Cart Items – Contains information on each product in the cart of a specific customer
- Transaction Point Type – Contains the classification names that a given transaction may start from or end to inside the business, eg M-Pesa, Bank
- In Business Transaction – Contains information on all the transactions that have been made within the business eg paying employees, moving funds from Bank to M-Pesa etc

Note: One can access all these tables in the website using the following format

- /tables/tablename

If you provide an incorrect table name you will be directed to the default product table page.

I did not include links to the tables in the webpage at this current time since I had not defined any error handling regarding updating information in the db as per the given constraints.

The following are the viable table names:

- user, customers, customer_payment_info, office_designation, office, employee_designation, employees, other_business_designation, other_business, product_package, product_types, products, inventory, reorder_status, reorders, order_status, orders, order_items, order_shipment, invoices, payment_methods, payments, direct_purchases, direct_purchase_items, product_images, image_designation, general_image, cart_items, transaction_point_type, in_business_transaction

Regarding the ERD

Having 30 tables in the ERD made the ERD quitefully big to be included in this documentation so I have it in the zip folder as well

I have the ERD both with the data types visible and not visible.The one with the data types visible is unfortunately clunky. You can actually generate the sql statements to create the tables on lucidchart if you have a premium membership, or if you are on the premium membership trial, as I have seen on youtube videos. This however did not work in my case for some reason.

The same can be said for mysql workbench, though I did not use mysql. I used postgresql.

Challenges Faced

The making of this project brought about the following challenges

- Media Queries working on resizable computer window but not on phones:
    - It happened that some parts of the media queries that I defined and tested on my computer do not apply to the small outdated ios phone that I have. I will have to research on that
- Colours on ios phones:
    - It also happens colours I defined look okay on the computer but not that accurate on ios devices as far as I have seen. That will also have to be looke on to
- Error handling:
    - Error handling is very necessary in keeping the application from crashing. However, it is for sure a very tedious task at times. It has to be done pretty much everywhere.
- Dynamically executing code:

    - It happens that at times whatever you want your program to execute depends on the values that you receive and on top of that you have say 100 different values in the least such that you cannot define keyworded variable arguments for the same. Exec and eval have really helped regarding this. But have been a pain as well.
- Web security:
    - I chose to host the web application using a linode server which brought about a number of security risks. One would be that every server has by default a root user with the username root who has sudo priviledges in changing everything regarding the application. Since I did everything using the root user that has the username of root, a hacker can easily ssh into my server after cracking my password, having known my username.

Future Work

The project has really brought forth a new perspective to things as well as yet more concerns that will be worked towards in the future. Some are as listed below:

- Applying restrictions to certain pages meant for managers, employees, etc
- Finishing up on the features ommitted to enable a user to add items to cart, place orders as well as make purchases to their orders during order placement or after order has been delivered
- Figure out how to accept payments via M-Pesa, Debit Card, Credit Card and any other online mode of payment
- Work on making the website more secure

Conclusion

Regarding the three objectives stipulated in the project description I achieved the following to some fair extent:

- I have been able to show a practical understanding of basic concepts related to Database Management Systems
- I have done my first project in the EEE course
- I have learnt to research on better ways to do things regarding programming

Appendix

Here is a sample of my code which I used to check if a given column is of type datetime. Using datetime.strptime you can conveniently convert a string to a datetime object as per a defined

formatting. This was of great help since html forms cannot quitefully do not send in data from forms in the form of datetime data type

```
if eval(f'type(record.{column}) is datetime'):
    exec(f'record.{column} = datetime.strptime(request.form["{column}"], "%Y-
%m-%d %H:%M:%S")')

else:
    exec(f'record.{column} = request.form["{column}"]')
```