# Quick Cart Blog

# Web Service

Version 1.0.0>

Prepared by

|  |  |  |
| --- | --- | --- |
| Dejeon Battick | 180914 | Dejeon.battick11@gmail.com |
| Shemar Lundy | 180916 | [lundyshemar@yahoo.com](mailto:lundyshemar@yahoo.com) |
| Shemar Henry  Mark Anthony | 180915 | [shemarhenry24@yahoo.com](mailto:shemarhenry24@yahoo.com) |
|  |  |  |

|  |  |
| --- | --- |
| Course Instructor: | Thomas Xu |
| Course: | Web and Mobile Application development II |
| Date: | June 23, 2019 |

# Overall Description

## Product Context and Need

The Quick Cart Blog was conceptualised with the average shopper in mind. The blog works alongside the Quick Cart mobile application for android and iOS and allows the user to have a sense of community. This community will allow users to share recipes with each other and be kept abreast on posts made by the Quick Cart blog administrator. These posts maybe but are not limited, sales, specials etc.

## Product Functionality

**Major functions of the system will be to:**

1. Allows the admin user to make posts to the blog.
2. Allows all users to add recipes to the blog.
3. Allows a user to delete their post and or recipe.
4. Allows a user to update their post and or recipes.
5. Allows a user view all their posts and or recipes.

**Non-functional requirements of the system will be to:**

1. Posts and recipes should be stored on a secured database.
2. User password should be hashed to ensure security.
3. System must check credentials against the user database to ensure that there isn’t duplication in the email field.
4. System should check to ensure that form fields are properly filled out.
5. System should ensure that a user is logged in before accessing certain features such as creating, updating and deleting posts and or recipes.
6. Should have separate dashboards for administrative users and regular users.

## Stakeholders and Users Characteristics

**The key stakeholders of this system are:**

**Shoppers:**

- Persons who use the Quick Cart application and also utilise the blog feature.

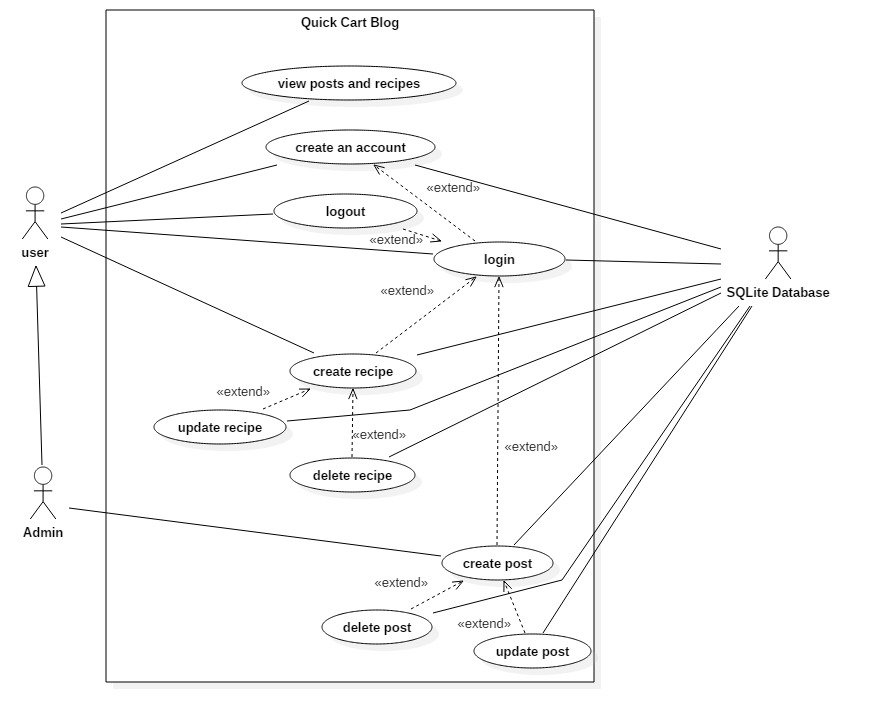
**Store Staff:**

- Persons who make posts on the behalf of the Store through the use of the administrative login.

## Operating Environment

Seeing as the blog is a Flask web service an internet connection is needed in order to access the site. The server on which the web service is running must be up and running and publicly accessible.

## Use Case Diagram



## Use Case Description

**Use case name:** view posts and recipes

**Summary:** the actor should be able to view all the posts and recipes that have been created.

**Primary Actor:** user, admin

**Secondary Actor:** n/a

**Precondition:** the user must access the site to view all the posts and recipes.

**Main Sequence:**

1. The user accesses the blog through use of the app.
2. The user is brought to the home page where they can view all the posts made.
3. The system adds the party to the database.

**Alternative sequence:**

**Step 2:** if the user clicks the recipe link, he/she will be brought to the recipe page where they can view all the recipes that have been added to the blog.

**Post-condition:** the posts and recipes added to the system have been successfully viewed by the user.

**Use case name:** create an account

**Summary:** the actor should be able to create an account for the blog.

**Primary Actor:** user, admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched and the register link was clicked.

**Main Sequence:**

1. The user fills out all the necessary field in the registration form.
2. The user clicks the submit button.
3. The site validates the information that has been submitted by the user.
4. The user is brought to the login page.

**Alternative sequence:**

**Step 3:** if the information submitted does not pass the validation the creation process will fail and the user will be prompted with the necessary error message(s).

**Post-condition:** the user has successfully created an account.

**Use case name:** login

**Summary:** the actor should be able to login to the site using the credentials they provide when creating the account.

**Primary Actor:** user, admin

**Precondition:** site was successfully launched and the user is on the login page.

**Main Sequence:**

1. User provides the required information.
2. The site validates the given information.
3. The site then brings the user to the home page and allows access to login specific features.

**Alternative sequence:**

**Step 2:** if the information submitted does not pass the validation check the login process will fail and the user will be prompted with the necessary error message(s).

**Post-condition:** the user was successfully logged into their account.

**Use case name:** logout

**Summary:** the actor should be able to logout to the site.

**Primary Actor:** user, admin

**Precondition:** the site was successfully launched and the user is currently logged in.

**Main Sequence:**

1. User clicks the logout button.
2. The site logs out the user and brings them back to the log in page.

**Post-condition:** the user was successfully logged out of their account.

**Use case name:** create recipe

**Summary:** the actor should be able to create a recipe post for the blog.

**Primary Actor:** user, admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched and the actor successfully logged in.

**Main Sequence:**

1. The actor fills out all the necessary field in the create recipe form.
2. The actor clicks the submit button.
3. The site validates the information that has been submitted by the actor.
4. The actor is brought to the recipe page.

**Alternative sequence:**

**Step 3:** if the information submitted does not pass the validation check the recipe posting process will fail and the user will be prompted with the necessary error message(s).

**Post-condition:** the user has successfully created a recipe post.

**Use case name:** delete recipe

**Summary:** the actor should be able to delete his/her recipe post for the blog.

**Primary Actor:** user, admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched, the user logged in and a recipe is exists in the database.

**Main Sequence:**

1. The actor goes to the account page.
2. The actor clicks the delete button for a recipe.
3. The recipe is deleted.

**Post-condition:** the user has successfully deleted their recipe post.

**Use case name:** update recipe

**Summary:** the actor should be able to update his/her recipe post for the blog.

**Primary Actor:** user, admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched, the user logged in and a recipe exists in the database.

**Main Sequence:**

1. The actor goes to the account page.
2. The actor clicks the update button for a recipe.
3. The actor is then brought to the update recipe page where the recipe information is displayed in the form.
4. The actor edits the displayed information.
5. The actor clicks the submit button.
6. The site does a validation check on the newly submitted information.
7. The actor is brought to the recipe page.

**Alternative sequence:**

**Step 6:** if the information submitted does not pass the validation check the recipe updating process will fail and the actor will be prompted with the necessary error message(s).

**Post-condition:** the actor has successfully updated their recipe post.

**Use case name:** create post

**Summary:** the actor should be able to create a post for the blog.

**Primary Actor:** admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched and the actor successfully logged in.

**Main Sequence:**

1. The actor fills out all the necessary field in the create post form.
2. The actor clicks the submit button.
3. The site validates the information that has been submitted by the actor.
4. The actor is brought to the home page.

**Alternative sequence:**

**Step 3:** if the information submitted does not pass the validation check the posting process will fail and the user will be prompted with the necessary error message(s).

**Post-condition:** the user has successfully created a post.

**Use case name:** delete post

**Summary:** the actor should be able to delete his/her post for the blog.

**Primary Actor:** admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched, the user logged in and a post exists in the database.

**Main Sequence:**

1. The actor goes to the account page.
2. The actor clicks the delete button for a post.
3. The post is deleted.

**Post-condition:** the user has successfully deleted their post.

**Use case name:** update post

**Summary:** the actor should be able to update his/her post for the blog.

**Primary Actor:** admin

**Secondary Actor:** database

**Precondition:** the site was successfully launched, the user logged in and a post exists in the database.

**Main Sequence:**

1. The actor goes to the account page.
2. The actor clicks the update button for a post.
3. The actor is then brought to the update post page where the post details are displayed in the form.
4. The actor edits the displayed information.
5. The actor clicks the submit button.
6. The site does a validation check on the newly submitted information.
7. The actor is brought to the home page.

**Alternative sequence:**

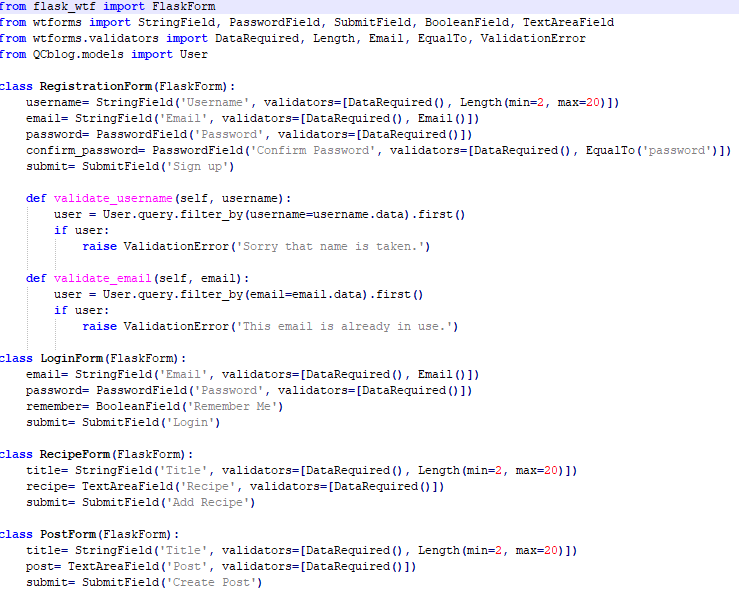
**Step 6:** if the information submitted does not pass the validation check the recipe updating process will fail and the actor will be prompted with the necessary error message(s).

**Post-condition:** the actor has successfully updated their recipe post.

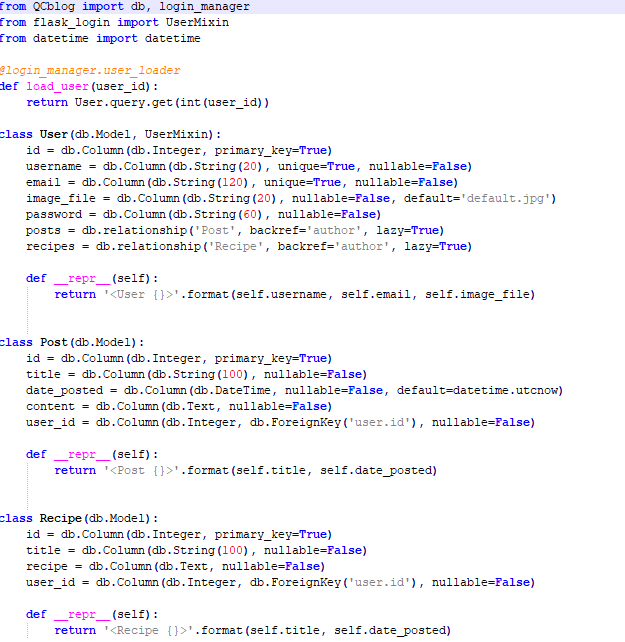
## Development

The Quick Cart Blog was developed using the Flask API. Flask is a restful API that is used in python that allows for developers to create web services that run on a server that was created in python. In implementing the functionality of the blog, several features of flask was utilised.

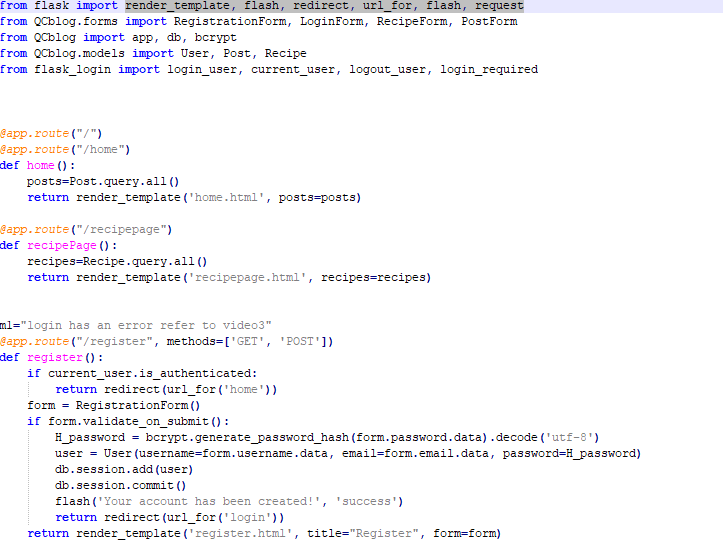
WTForms were used to create the forms that captured user input and added them to the database. Along with these forms Validators were used to ensure that user input was appropriate and did not contain anything that would be fatal to the web service.



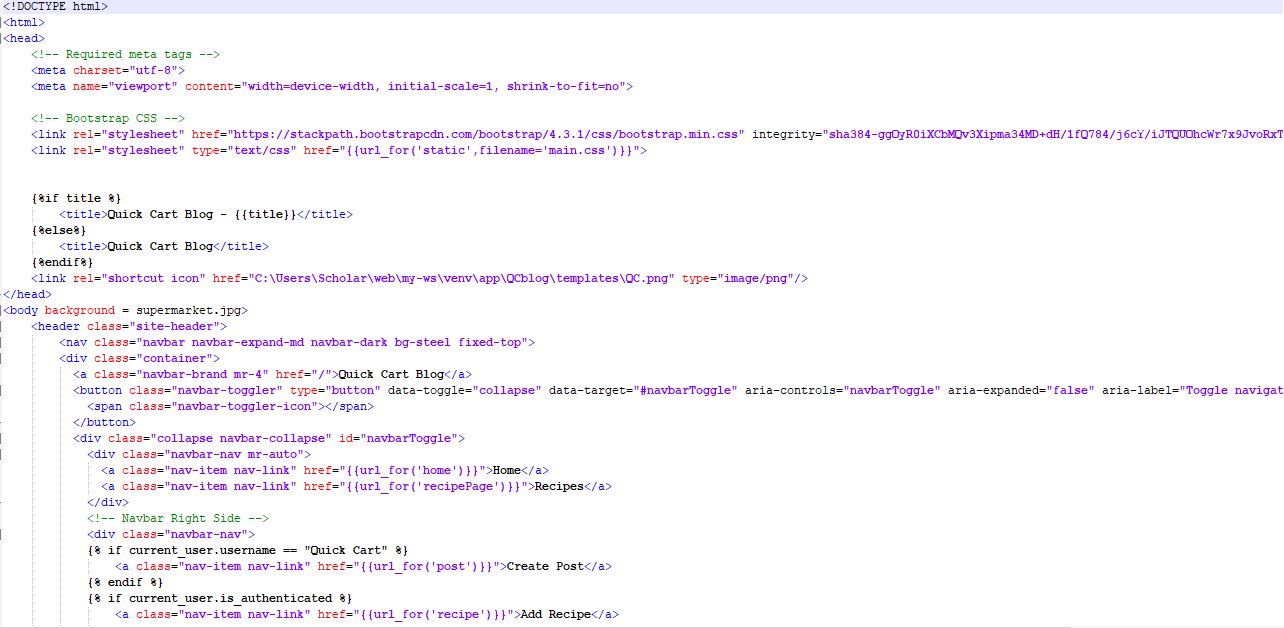
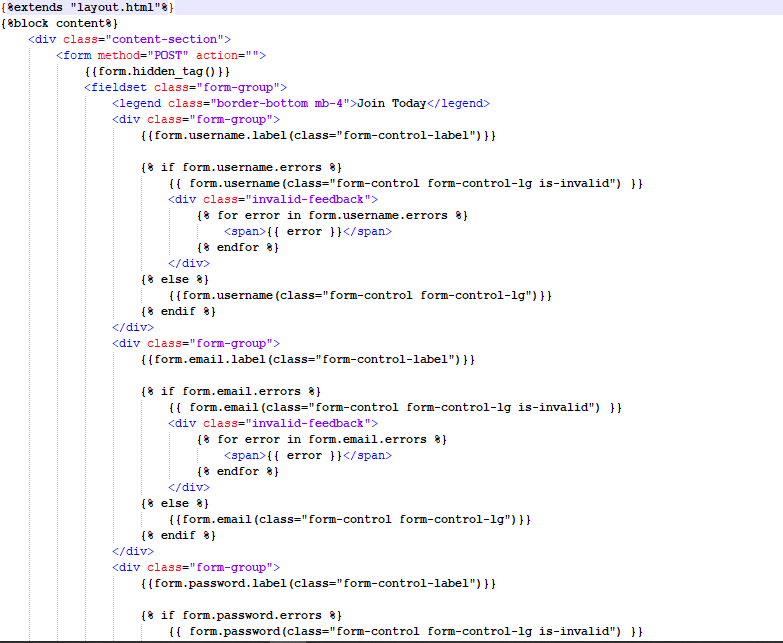
The database was created using SQLAlchemy which always flask to communicate with databases that can be utilised by the web service.



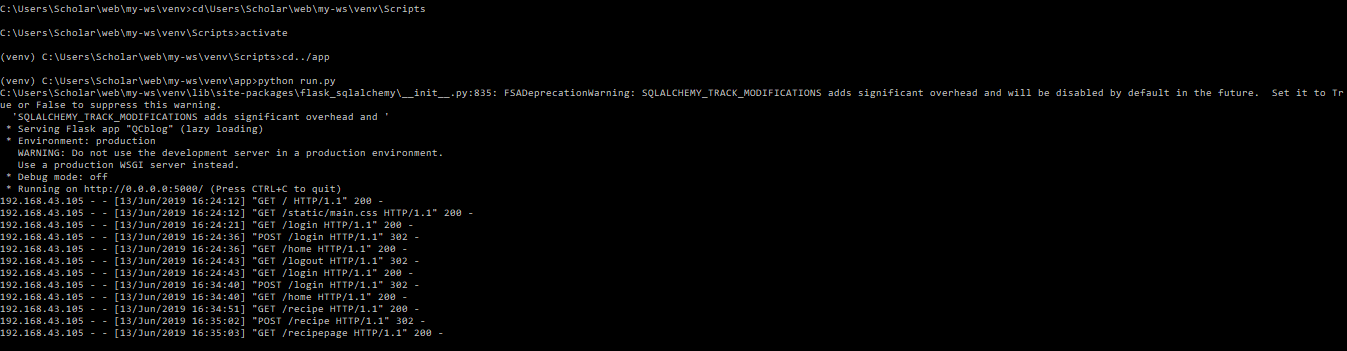
Flask features such as flask\_login, render\_template, flash, redirect etc. were also utilised to add more functionality to the web service. The flask\_login functionality allows for the service to carry out certain functions such as using the information of the current user, ensuring that a page can only be accessed if the user is logged in and also allows for users to logout. The render feature allows for html templets to be rendered. Flash is used to display message to the user. Redirect is used to move the webpage from one page to another using the app route. Bcrypt was also utalised during development to allow the hashing of passwords.



The development of the website was done using HTML, CSS and Bootstrap. HTML (hypertext markup language) was used to create the content for website which was then styled using CSS and Bootstrap templates.



After the completion of the coding aspect of the project we then launched the application using the command line thus launching the server.



As seen in the above picture, when the site is launched it utilises GET and POST methods in order to retrieve data from the database and handle data being submitted by the user.