Lavender Dangerous

Overview

weBulk is an application designed to provide users with bulk pricing discounts without the need to buy in large quantities. It works by coordinating purchase requests across many different users so that bulk pricing is achieved and then divides the total cost across the pool of buyers depending on their requested amount of product. Our web application allows users to save money by taking advantage of discounts that wouldn't otherwise be realistically available to them.

Team Members

Alex Willinder: awillinder@umass.edu
Thomas Palaschak: tpalaschak@umass.edu
Nishant Nawathe: nnawathe@umass.edu
William Warner: wwarner@umass.edu
Jake Celentano: jcelentano@umass.edu
Ajibola Famuyibo: afamuyibo@umass.edu

Github Repository

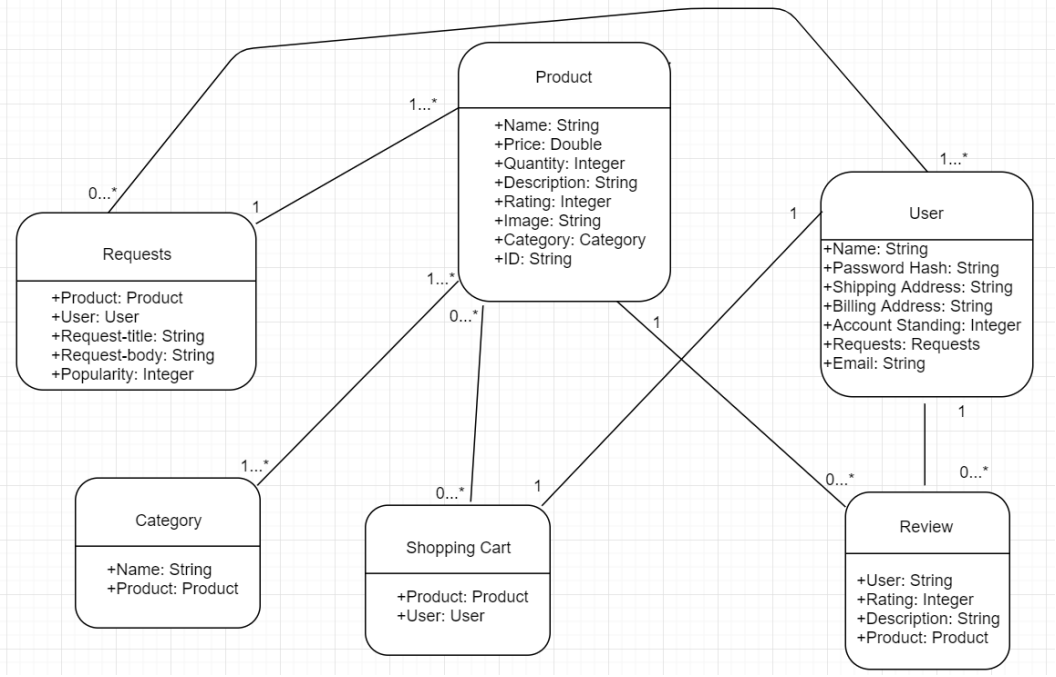
https://github.com/nnawathe/Lavender-Dangerous

Design Overview

In order to implement our web application effectively, we created six different models to represent information. The Product model defines each of our products in the database: each product is assigned a unique ID (primary key), as well as name, quantity, price, and description attributes. Category designates a category for each product. User is the model to represent user accounts in our application, and consists of a name, email, password hash, billing and shipping address, and account standing (a measure of how trustworthy this user is when following through requests). Shopping is a one-to-one field connected to user and consists of the items the user is intending to buy. Requests consist of a user field that is a foreign key to the User model (every request must be from a user), in addition to a title, text describing the request, and the popularity of the request.

weBulk uses six URL routes to navigate users through each part of our application. In addition to the homepage, there is the Products, Account, Cart, FAQ, and Requests page. FAQ is the only page that is not connected with the others in some way, as it only provides a static view of common questions and answers. Products directs to a view detailing the products populated from our database. Requests directs to a view of requests made by users of the application, including which ones are popular and the number of requests for items. Account displays the User's home page, with account settings, their requests, and popular products for them. Cart is connected to Account and displays the products a user intends to buy.

Data Model Diagram



Problems/Successes

Communication was a strong point for our team in this phase of the project: we were able to meet both inside and outside of class to discuss and flesh out our relational model. In addition, we used a Messenger group to stay in contact regarding the project and answer each other's questions as needed. This allowed us to stay on track throughout the development of our project and kept each group member on the same page regarding their responsibilities.

The implementation of our project was straightforward. Our team utilized GitHub effectively, and in doing so avoided any conflicts that could complicate our project design or cause headache-inducing bugs. Database implementation was straightforward, and with the help of the Django tutorials we were able to implement our project with no issues.

Our team coordinated well on this phase of the project by communicating constantly and partitioning the work up in a way that allowed every person to contribute to the project. We ran into no team problems in this part of the project. As the development of our web application increases in complexity it would likely be wise to schedule more team meetings. This would help ensure that everyone remains on the same page and would be conducive to tackling more complex parts of the project in the future.

Individual Write-up

Nishant Nawathe

For the project I first helped by starting up the initial Django database setup, which was then finished by someone else. I also made the requests page, which involved the taking of data from the database and adding it to the requests page through views.py and the HTML side as well. I also adjusted FAQ to work with the new basic_generic.html template. I did about 15% of the work out of the whole project.

Thomas Palaschak

I added the empty template pages for each of the templates/views we needed. I created our urls.py file and configured the urls for each of our views accordingly. I created and completed the accounts template so that it can populate data as needed. Finally, I created and wrote each part of the main write-up. Overall, I did about 19% of the work out of the whole project.

William Warner

I wrote and tested the models.py implementation by adding some mock data. In addition, I created template views.py with method stubs, and created the base_generic.html template, the index.html file and view, and the static content directory. I think I did about 25% of the work.

Jake Celentano

For our second project, I helped with the initial Django setup by providing some detailed views for our admin pages. I also debugged a few of the python scripts for some minor errors. I think I did around 12% of the work for this section.

Alex Willinder

I added a good number of products into our mock data (I think around 15), and collected and store images for each of them. I also took care of the product and cart templates, for which I had to work with and manipulate the data models for product, shopping cart, and review. I think I'm also tagging and submitting, but that won't be done until after this document it finished. I think I did about 17% of the work.

Jay Famuyibo

I made the several versions of the data model diagram, including the final version, and assisted in the project organization and logistics. I did about 12% of the work.