Detailed project specification

Team 27

Qingyi Wang(qingyiwa)
Xinze Zhou(xinzez)
Siying Jin(siyingj)

Original Proposal

Zero Touch Full-stack in a Few Clicks
An end-to-end automated website builder and generic zero-touch web hosting framework for small-medium scale business owners to host their businesses or prototype their ideas within a few minutes. A super-fast and easy way to launch their business websites not worrying about storage, resilience and security.
Our features include:

1. Provide a text-based automatic generic website template generation.
2. Naming service/URL domain name provisioning.
3. Complete AWS/cloud integration with secure storage.
4. Utilize click rate, customer insights, time spent on page and other metrics to improve page popularity
5. Design a set of heuristics that help determine what keywords/terms that could be included to improve the popularity of their business.
6. Seamless integration with Yelp to further popularize the business.
7. Have a connected platform to host ads, videos on their products (could be as simple as a dedicated html div component in the framework) and share ad feeds among other registered business owners and users.

Feedback:
To maximize your chance of success, we recommend:
* A small case study of similar app, like Shopify, would help you with design decisions.
* Focus on completing the main features (1,2,4)
* Trimming down integration with other services (AWS, Yelp, etc) into one or two. Start with a simple one first, like Yelp where you have to just link it to appropriate APIs. AWS might be bit complicated because you need to do things like get user's credentials.

Our Modifications:

We're going to build an end-to-end website builder and management system for retailers and small businesses. We will focus on 1,2,4 in the above specification. Users can use this web app to create their own website and monitor the orders and products.

1. Template:
Instead of building a generic website template, we will design several templates with basic layout, and each of them will consist of necessary website functionalities(including homepage, shopping cart, collection list..) for users to choose from.

2. Domain:
We will allow the user to provide their own shop name and display the name at the corresponding position of the webpages to provide this site with a unified look;
We will use the url pattern tools in django web framework to provide a feature of allowing the user to customize their own shop url.

3. Optimization on attracting customers:
We will also categorize each product and make recommendations based on the previous purchased product's category.
In addition, we will build filters based on a product's price, category and number of products available to buy.

4. Data input:
We will enable the user to upload a excel/csv file as data sources; we may also enable them to import data from another CRM platform, such as Salesforce.

Technologies

Languages: Python, HTML, CSS, Javascript
Frameworks: Django, Bootstrap, Ajax
APIs: Salesforce API

Functionalities

(CATEGORIZED BY FEATURES)

- ❏ Create User & User Authentication (Xinze)
  To be specific, here we split it into two parts:
  1. Allow the owner of the store to register, log in and organize their store settings in their user portal, see below.
  2. Allow the customers to register and log in. Each of the customer will have his/her own shopping cart. They are also able to view their history orders.

- ❏ User Portal for Shop Owners: (Xinze all but 3)
  1. Edit profile/account settings.
  2. View product list(items in stock/purchased), add/remove/edit product, add product from external storage.
  3. View/edit orders in a given period, add orders from external storage. (Siying)
  4. Analytics of orders: reports of total sales, live view of sales, find top products based on click rate and order history.
  5. Analytics of customers: search history, purchase history, product, recommendation for customers based on favorite categories.
  6. View ongoing activities and real-time preview of customizing a website.
  7. A link to create website, see below.

- ❏ Create Website: (Qingyi will focus on front-end of this part, Siying will work on the rest)
  1. Customize the url using django.conf.urls tools.
  2. User can choose a pre-defined template and customize the template: Change colors of the features of your website, change fonts, customize favicon, add and delete the existing sections of the page, edit the title, subheadings and content.
  3. Preview the site and save changes.

- ❏ The Shop : (Qingyi)
  Renders all the features that have been customized by the user .
  1. The shop homepage which provides a gallery of the products and their price. The customer will be able to skim through all the products and filter their target items by using the built-in filter.
  2. Product page with the detailed information and price of the product
  3. View the shopping cart
  4. Checkout page

  And provides effective links between these pages, thus providing a complete, functional shop website for the customers.

Data Models

User:
- username: TextField
- password: TextField
- first name: TextField
- last name: TextField
- photo: ImageField

Shop:
- user: ForeignKey(‘User’)
- name: TextField

Theme(Enum) {Black&White,.. }

Website:
- shop: ForeignKey(‘Shop’)
- base_url: TextField
- template: Theme

PageType(Enum) {Homepage, Cart, Collection...}

Page:
- type: PageType
- color: IntegerField
- font: TextField
- url: TextField

Section:
- Page: ForeignKey(‘Page’)
- visible: BooleanField
- position: ArrayField
- text: TextField
- description: TextField
- options: ArrayField

Customer:
- shop: ForeignKey(‘Shop’)
- username: TextField
- password: TextField

Collection: {eg, Men, Women, Face, Body..}

name: TextField
description:TextField

Product:
shop: ForeignKey(‘Shop’)
name: TextField
description: TextField
price: FloatField
image: ImageField
collections: ManyToManyField(‘Collection’)
inventory: IntegerField

Order:
customer: ForeignKey(‘Customer’)
shop_id: IntegerField
products: ManyToManyField(‘Product’)
total price: FloatField
time: DateField