BHF Inventory Management

Capstone Project Plan

Attila Farkas

Table Of Contents

Table Of Contents
Project Overview
Project Goals
Example Usage and Data
Technical Approach
Scope and Timeline of Capstone Project
User Stories
    User Account Management and User Authentication
    Inventory Management
        Products
        Categories and Subcategories
    Webstore Listings
Entity Relationship Diagram
UML Diagram
Project Structure
Next Steps

Project Overview

I am making a website that provides an administrative interface to build and maintain an inventory of bedding, quilts, duvet covers and similar home decor products. Additionally, this website allows creating, querying, removing and updating listings of inventory items in two online retail stores: eBay storefront, and CS-Cart-based webstore (http://www.bellahomefashions.com).

The proprietor, who is currently operating both of these online retail stores, will benefit from using this website by maintaining a common inventory of products for managing all online retail listings. This solution will provide consistency of product listings across all retail sites, and streamline the management of these listings.

Project Goals

The BHF Inventory Management website has the following main goals:

- creation and maintenance of product inventory database for a business that sells home decoration products
- management (CRUD) of sales listings of inventory items on both http://www.bellahomefashions.com, and on the proprietor's eBay storefront

Example Usage and Data

Create a database of product inventory. The following key data will be stored:

- product name and descriptions
- product available colors and sizes
- product tracking codes
- product pricing info
- product supplier address and contact info
- product quantity on hand

- product listing category and subcategory info
- images of products, categories and subcategories
- metadata for product listings
- product and listing status

Technical Approach

The website will be running on the Django 1.8 framework with a PostgreSQL database. Django's built-in authentication functionality will be used for user authentication to control website access. Django's admin interface will be used to manage user accounts. Bootstrap and jQuery will be used to implement the front end.

Scope and Timeline of Capstone Project

A minimum viable product for the Capstone Project will include the following:

- provide user account management via the Django admin interface
- provide user authentication (username/password) in order to access website
- create, list, modify and deactivate products
- product inventory listings, optionally filtered on common product attributes such as category/subcategory, tracking codes, and status
- create, list, modify and deactivate categories and subcategories
- provide uploading of images and thumbnails, which may be associated with inventory products, categories and subcategories

Optional features:

- search products, categories and subcategories on words in textual fields
- add dashboard homepage with statistics on inventory and webstore listings

The following features are outside of the scope of the Capstone Project, and may be implemented subsequently:

- CRUD functionality for managing web store listings

Capstone Project to be completed by November 9th.

User Stories

The Inventory Management user stories will be implemented by the completion of the Capstone Project. The Web Store Listings user stories are outside of the scope of the Capstone Project and are preliminary in their current state.

User Account Management and User Authentication

- The Django admin interface, after logging into it, shall be used to add, modify and delete user accounts in the built-in User table.
- A user loading any page of the website will be required to log in before any webpage on the site is accessible. Thereafter the website is accessible until the user logs out.
- A user may log out of the website using the logout button on any inventory page.

Inventory Management

Products

- The Products navigation button loads the Product List page. It lists all existing products, with thumbnail image, internal code, product name and status properties displayed. The list of products may be filtered on category/subcategory, tracking codes, and status. The status property for a product may be changed in this list by selecting the desired one from a dropdown list.
- All of a product's associated properties may be viewed or updated by following the product name link. Product properties supplier, color, size, status, image may be selected from existing lists, each of which may be edited in a popup window from the Edit Product page.

- A new product may be created using an add button on the product list page. Editing the list of selectable properties is also available via popup windows from the New Product page.

Categories and Subcategories

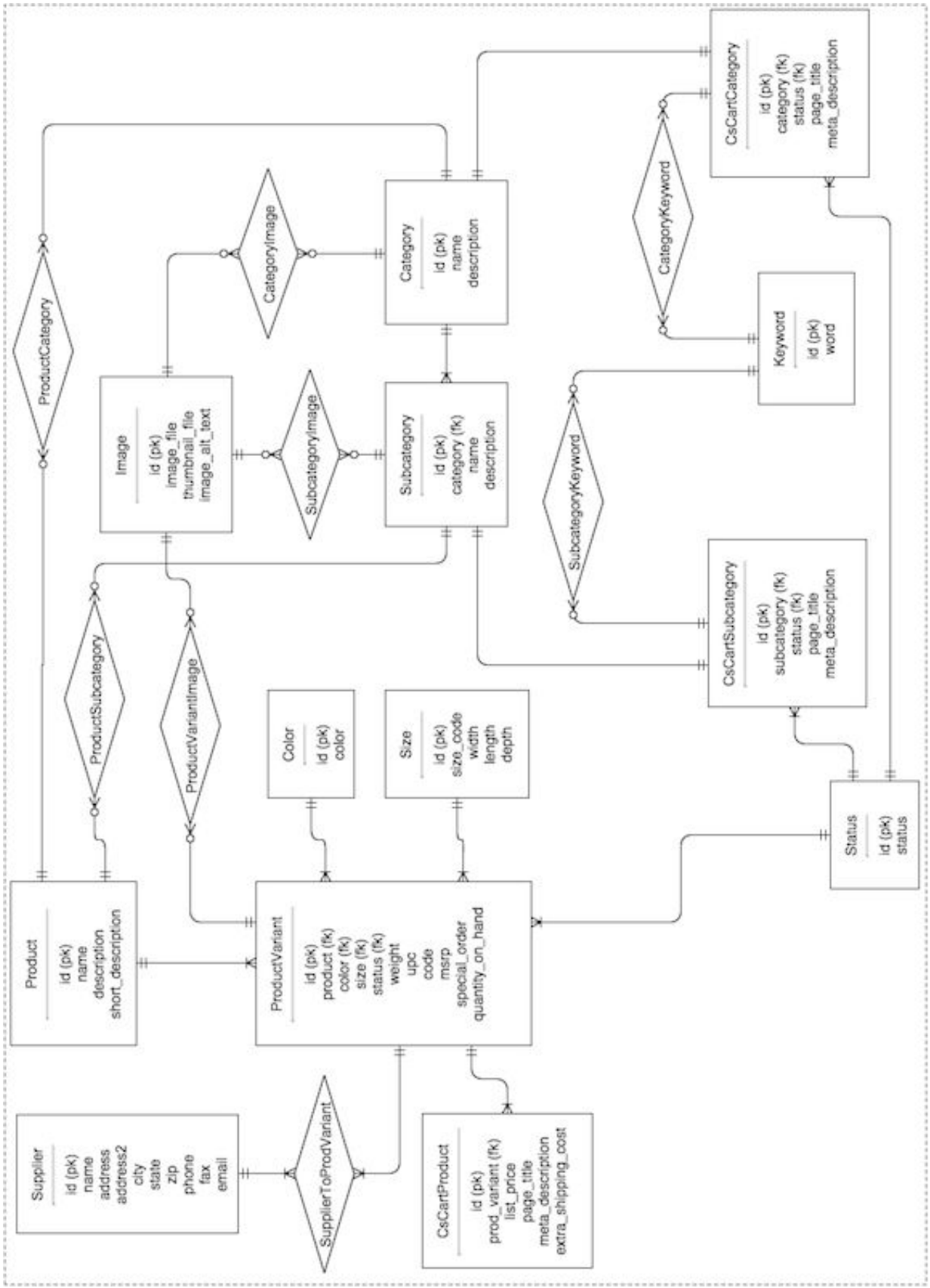
- The Categories navigation button loads the Categories List page. It lists all existing categories, with thumbnail image, category name and status properties displayed. The status property for a category may be changed in this list by selecting the desired one from a dropdown list. Clicking the subcategory button for a category loads the Subcategory List page which lists all associated subcategories, with thumbnail image, subcategory name and status properties displayed.
- All of a category's or subcategory's associated properties may be viewed or updated by following the category name or subcategory name link, respectively. The image property may be selected from the existing list, which may be edited in a popup window from the Edit Category or Edit Subcategory page.
- A new category or subcategory may be created using an add button on the respective list page. Editing the list of selectable properties is also available via popup windows from the New Category or New Subcategory page.

Webstore Listings

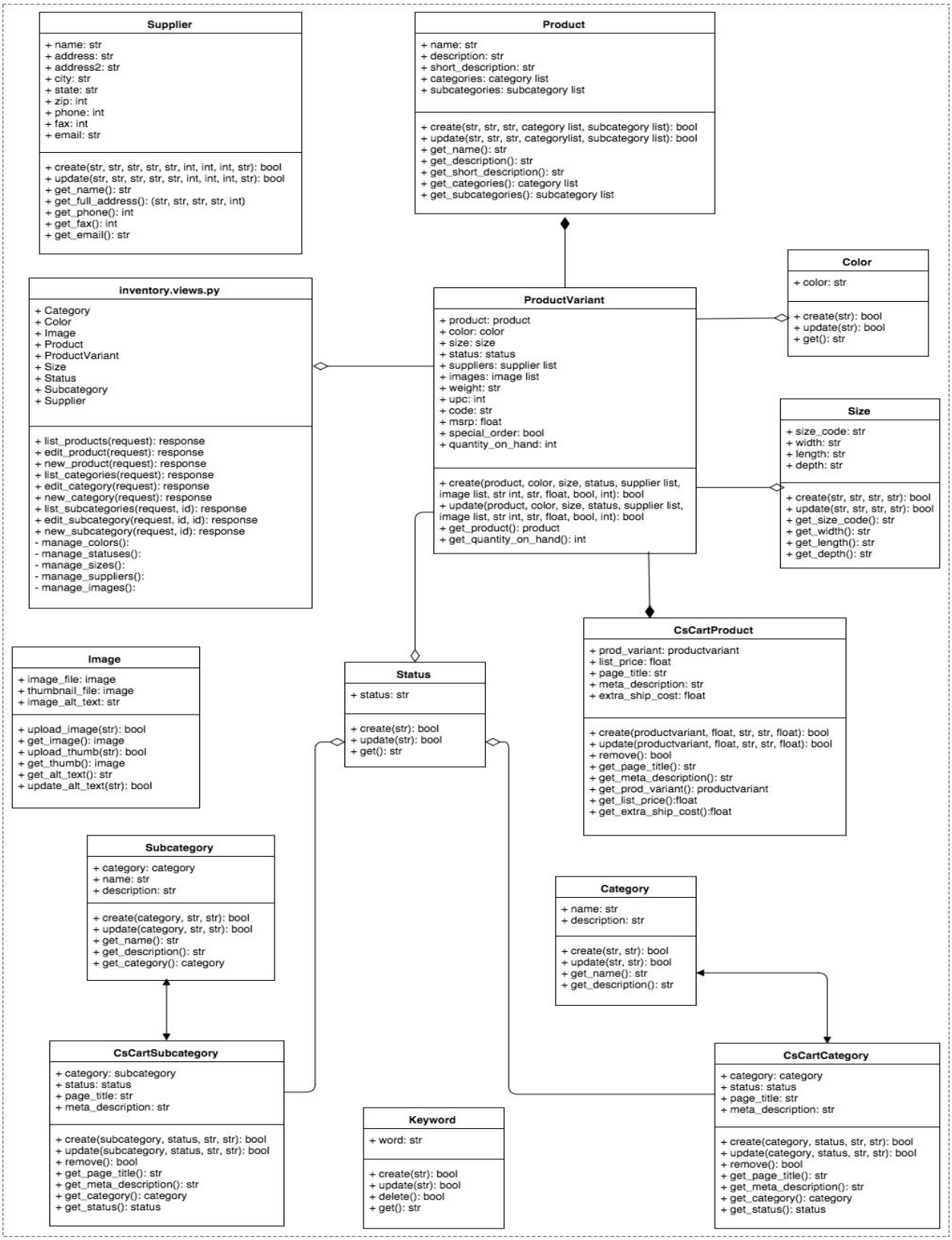
- The user may select either Bella Home Fashions or eBay store using the Listings navbar button. Each selection navigates to its corresponding page with a common set of user stories outlined below.
- The user may display a complete or filtered list of products in the webstore. Each product listing has a set of properties, such as listing page title, description, list price, available quantity, and extra shipping cost, which may be edited and the updated listing uploaded to the webstore.
- The user may create a new listing from an Inventory Product and upload it to the webstore.

- The user may display a list of categories and subcategories currently in the webstore. User may upload categories and subcategories from the Inventory to the webstore.
- User may customize category and subcategory metadata, such as keywords, description, webstore page title, and upload this metadata to the webstore.

Entity Relationship Diagram



UML Diagram


Supplier
+ name: str
+ address: str
+ address2: str
+ city: str
+ state: str
+ zip: int
+ phone: int
+ fax: int
+ email: str
+ create(str, str, str, str, str, int, int, int, str): bool
+ update(str, str, str, str, str, int, int, int, str): bool
+ get_name(): str
+ get_full_address(): (str, str, str, str, int)
+ get_phone(): int
+ get_fax(): int
+ get_email(): str
Product
+ name: str
+ description: str
+ short_description: str
+ categories: category list
+ subcategories: subcategory list
+ create(str, str, str, category list, subcategory list): bool
+ update(str, str, str, categorylist, subcategory list): bool
+ get_name(): str
+ get_description(): str
+ get_short_description(): str
+ get_categories(): category list
+ get_subcategories(): subcategory list
Color
+ color: str
+ create(str): bool
+ update(str): bool
+ get(): str
inventory.views.py
+ Category
+ Color
+ Image
+ Product
+ ProductVariant
+ Size
+ Status
+ Subcategory
+ Supplier
+ list_products(request): response
+ edit_product(request): response
+ new_product(request): response
+ list_categories(request): response
+ edit_category(request): response
+ new_category(request): response
+ list_subcategories(request, id): response
+ edit_subcategory(request, id, id): response
+ new_subcategory(request, id): response
- manage_colors():
- manage_statuses():
- manage_sizes():
- manage_suppliers():
- manage_images():
ProductVariant
+ product: product
+ color: color
+ size: size
+ status: status
+ suppliers: supplier list
+ images: image list
+ weight: str
+ upc: int
+ code: str
+ msrp: float
+ special_order: bool
+ quantity_on_hand: int
+ create(product, color, size, status, supplier list, image list, str int, str, float, bool, int): bool
+ update(product, color, size, status, supplier list, image list, str int, str, float, bool, int): bool
+ get_product(): product
+ get_quantity_on_hand(): int
Size
+ size_code: str
+ width: str
+ length: str
+ depth: str
+ create(str, str, str, str): bool
+ update(str, str, str, str): bool
+ get_size_code(): str
+ get_width(): str
+ get_length(): str
+ get_depth(): str
Image
+ image_file: image
+ thumbnail_file: image
+ image_alt_text: str
+ upload_image(str): bool
+ get_image(): image
+ upload_thumb(str): bool
+ get_thumb(): image
+ get_alt_text(): str
+ update_alt_text(str): bool
Status
+ status: str
+ create(str): bool
+ update(str): bool
+ get(): str
CsCartProduct
+ prod_variant: productvariant
+ list_price: float
+ page_title: str
+ meta_description: str
+ extra_ship_cost: float
+ create(productvariant, float, str, str, float): bool
+ update(productvariant, float, str, str, float): bool
+ remove(): bool
+ get_page_title(): str
+ get_meta_description(): str
+ get_prod_variant(): productvariant
+ get_list_price():float
+ get_extra_ship_cost():float
Subcategory
+ category: category
+ name: str
+ description: str
+ create(category, str, str): bool
+ update(category, str, str): bool
+ get_name(): str
+ get_description(): str
+ get_category(): category
Category
+ name: str
+ description: str
+ create(str, str): bool
+ update(str, str): bool
+ get_name(): str
+ get_description(): str
CsCartSubcategory
+ category: subcategory
+ status: status
+ page_title: str
+ meta_description: str
+ create(subcategory, status, str, str): bool
+ update(subcategory, status, str, str): bool
+ remove(): bool
+ get_page_title(): str
+ get_meta_description(): str
+ get_category(): category
+ get_status(): status
Keyword
+ word: str
+ create(str): bool
+ update(str): bool
+ delete(): bool
+ get(): str
CsCartCategory
+ category: category
+ status: status
+ page_title: str
+ meta_description: str
+ create(category, status, str, str): bool
+ update(category, status, str, str): bool
+ remove(): bool
+ get_page_title(): str
+ get_meta_description(): str
+ get_category(): category
+ get_status(): status

Project Structure

- bhf (project configuration)
- cs_cart (app)
    - migrations
    - static
        - cs_cart
            - css
            - js
    - templates
- inventory (app)
    - migrations
    - static
        - inventory
            - css
            - js
    - templates
        - registration (authentication)
- static
    - css (site wide)
    - js (site wide)
- media (user upload)
    - img

Next Steps

- Write unit tests for existing code
- Add Edit Product and New Product pages
- Add thumbnails and full size images to the product, category, subcategory edit pages
- Add popup window to add and edit entries of the following attributes:
    - image
    - supplier
    - color
    - size
    - status
    - keyword
- Add form styling on edit pages
- Add product list filtering on category/subcategory
- Add product search on internal product code
- Add pulldowns for changing product status on product list page, including AJAX functionality to update database
- Add breadcrumbs or similar site navigation
- Add pagination to Product, Category and Subcategory list pages
- Add dashboard homepage
- Add CS-Cart CRUD functionality
- Add eBay CRUD functionality