READ ME

GitHub
https://github.com/ameyaam/project

Pivotal Tracker
https://www.pivotaltracker.com/n/projects/1310218

Models

FarmersMarket

Why this model?
Our project deals with bringing farmers markets online and for storing information about the farmers market we have created this model

Why this association?
has_many :products -> All products are related to atleast one farmer market and a market may contain many products.

Why these Validations?
City, State, Street and Name should be present for farmers market because these are the required details to display farmers market to the user.

Cart

Why this model?
This model is to hold information about the products a particular user wants to purchase while shopping on the site.

Why this association?
belongs_to :user -> each cart is related to one user, who is shopping on the site.
has_many :products -> each user cart can hold any number of products that the user wants to buy
has_many :farmers_markets -> each user can shop products from different markets.

Why these Validations?
User should be present because a cart belongs to a user and is of no use without user.

user

Why this model?
This model is used to store login user information.

Product

Why this model?
This model holds information about all the products that are present in all farmers markets. This involves product image, id and price of it

Why this association?
belongs_to :farmers_markets -> each product is uniquely associated with one farmer market and is available only at that place.
belongs_to :category -> each product is also associated with one category, this would be useful while the user is shopping. User can choose and browse through products belonging to one category.

Why these Validations?
Name should be present to display product to the user.
Price should be numerical and greater than 0 as implies for a product to be sold.
Category and Farmers market foreign key constraints must be satisfied as product depends upon them.

Category

Why this model?
Products can be of different categories and we have normalized the database to include category as separate model.

Why this association?
has_many :products -> category holds all the products that belongs to it. Useful to display products belonging to that particular category to the user.

Why these Validations?
Name has to be present because logically it is a primary key.

Controllers

farmers_markets_controller

#get_nearby_markets
This controller responds to the ajax request made by the front page. This responds with all the nearby markets, when the user selects a city.

Why this functionality in this controller?
get_nearby_markets makes use of the farmers_market model and hence is present in the farmers_markets controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to find nearby markets and continue shopping in them. We would like to implement appropriate view for this in the next iteration.

#set_market
This controller sets the market id as a session variable to user, so that it can be used to display items related to the market in further pages.

Why this functionality in this controller?
set_market makes logically belongs to this controller

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to find items and continue shopping.

categories_controllers

#get_all_categories
This controller responds to the ajax request made by the page to display all categories in the farmers market. This responds with all the categories, when the user selects a farmers market, all the categories in the market will be displayed to the user.

Why this functionality in this controller?
get_all_categories makes use of the categories model and hence is present in the categories controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to find particular products in the category and continue shopping. We would like to implement appropriate view for this in the next iteration.

Products_controllers

#get_all_products
This controller responds to the ajax request made by the page to display all products in the farmers market. This responds with all the products, when the user selects a farmers market, all the products in the market will be displayed to the user.

Why this functionality in this controller?
get_all_products makes use of the products model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to find and browse through products in the category and continue shopping. We would like to implement appropriate view for this in the next iteration.

#get_products
This controller responds to the ajax request made by the page to display products belonging to a particular category selected within the farmers market. This responds with all the products belonging to that category in the farmers market, when the user selects a category, all the products of it in the market will be displayed to the user.

Why this functionality in this controller?
get_products makes use of the products model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to find and browse through products in the category and continue shopping. We would like to implement appropriate view for this in the next iteration.

Carts_Controller

#add_to_cart

This controller responds to the ajax request made by the user to add a product to his cart. This controller retrieves the user session and adds the product id to his cart.

Why this functionality in this controller?
add_to_cart makes use of the carts model and hence is present in the carts controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the user to add items to the shopping cart. We would like to implement appropriate view for this in the next iteration.

Controllers generated through Scaffold

Products_Controller

#new

This controller responds to the request made by the page to add new product in the farmers market. This responds with a form to fill in details. when the admin fills in details about the new product and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the products model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new products to the market. We would like to use this view for the next iteration.

#create

This controller responds to the request made by the page to add new product in the farmers market. This inserts the new product into the database, when the admin fills in details about the new product and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the products model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new products to the market. We would like to use this view for the next iteration

#destroy

This controller responds to the request made by the page to destroya product in the farmers market. This deletes the product from the database, when the admin selects the product to be deleted and submits a request, it gets deleted from the database.

Why this functionality in this controller?
destroy makes use of the products model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to delete products from the market. We would like to use this view for the next iteration.

farmers_markets_controller

#new
This controller responds to the request made by the page to add new farmers market in the database. This responds with a form to fill in details. when the admin fills in details about the new market and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the farmers_markets model and hence is present in the farmers_markets controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new markets to the market. We would like to use this view for the next iteration.

#create

This controller responds to the request made by the page to create a new market in the database. This inserts the new market into the database, when the admin fills in details about the new market and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the farmers_markets model and hence is present in the products controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new markets to the database. We would like to use this view for the next iteration.

#destroy

This controller responds to the request made by the page to destroy a market in the database. This deletes the market from the database, when the admin selects the market to be deleted and submits a request, it gets deleted from the database.

Why this functionality in this controller?
destroy makes use of the farmers_market model and hence is present in the farmers_market controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to delete market from the database. We would like to use this view for the next iteration.

category_controller

#new
This controller responds to the request made by the page to add new category in the database. This responds with a form to fill in details. when the admin fills in details about the new category and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the category model and hence is present in the category controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new categories to the market. We would like to use this view for the next iteration.

#create

This controller responds to the request made by the page to create a new category in the database. This inserts the new category into the database, when the admin fills in details about the new category and submits a request, it gets added to the database.

Why this functionality in this controller?
new makes use of the category model and hence is present in the category controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to add new categories to the database. We would like to use this view for the next iteration.

#destroy

This controller responds to the request made by the page to destroy a category in the database. This deletes the market from the database, when the admin selects the category to be deleted and submits a request, it gets deleted from the database.

Why this functionality in this controller?
destroy makes use of the category model and hence is present in the category controller.

Why this functionality in this iteration versus the next iteration?
This serves as the back end function for enabling the admin to delete categories from the database. We would like to use this view for the next iteration.