**Christopher Romo**

**Fido Hub Documentation**

**Introduction (From Interview Worksheet)**

The application that I will be creating is a companion app to a dog-walking application I am creating in my UX/UI Class, “Fido” (Planned to be a mobile app). This app, “Fido Hub” (Website), will allow users to log in, determine if they are a client or an employee, and if they are a customer, register dogs. The primary purpose of this app is to allow customers to view information about employees, and allow employees to get information about a client and their dogs. It is kind of like social media, where users can see other users and add their own information, making connections between employees and clients easier. The difference between this app, “Fido Hub”, and the other application, “Fido” (UX/UI), is that “Fido Hub” will be mainly for information keeping, such as contact information for employees and clients, a section for important information about a dog, etc. This application, “Fido Hub”, will be like a social media website for employee and client connections. It will provide a way for customers to find their perfect employee, and for employees to learn more about their clients and their dogs.

**Part 1 – Kanban Board and User Stories**

For Kanban Board: <https://github.com/klafari/Fido-Hub.git>

**User Story 1 (Implemented in Sprint 01):**

As a client, I would like to register all of my dog’s information (like name, weigh, breed, etc.), so that future handlers can better understand my dog at a glimpse.

**User Story 2 (Next Iteration):**

As a client, I would like to be able to see the pages of future handlers, so that I can choose the perfect fit for me and my dog.

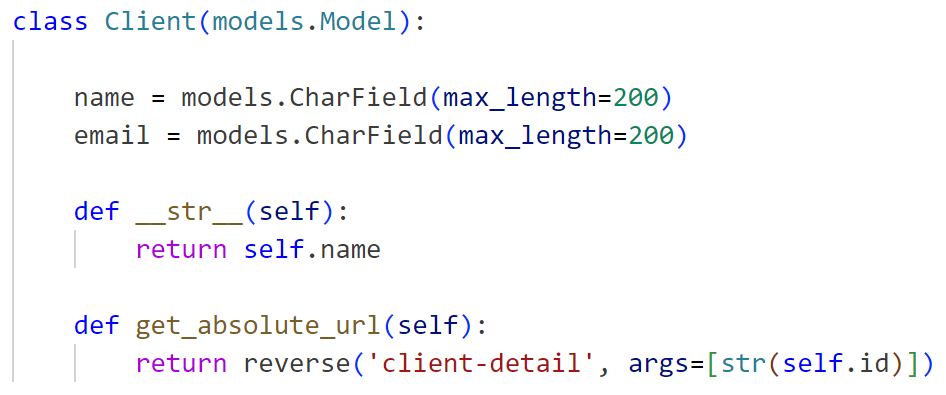
**User Story 3 (Next Iteration)**

As an employee, I want to be able to see clients and their dogs as detailed screens, so that I can do my research before reporting to work.

**Part 2 – MVT Relationships**

**Model 1: The Client**

The client is a general user of the website, they have the ability to register as many dogs as they want. They can also view Dog Walker pages and edit their own page. They have their own list view, and they will be able to upload pictures of their dogs and even submit profile pictures.



They have these associated views:

**ClientListView**

Template: client\_list.html

URL Namespace: ‘clients’

**ClientDetailView**

Template: client\_detail.html

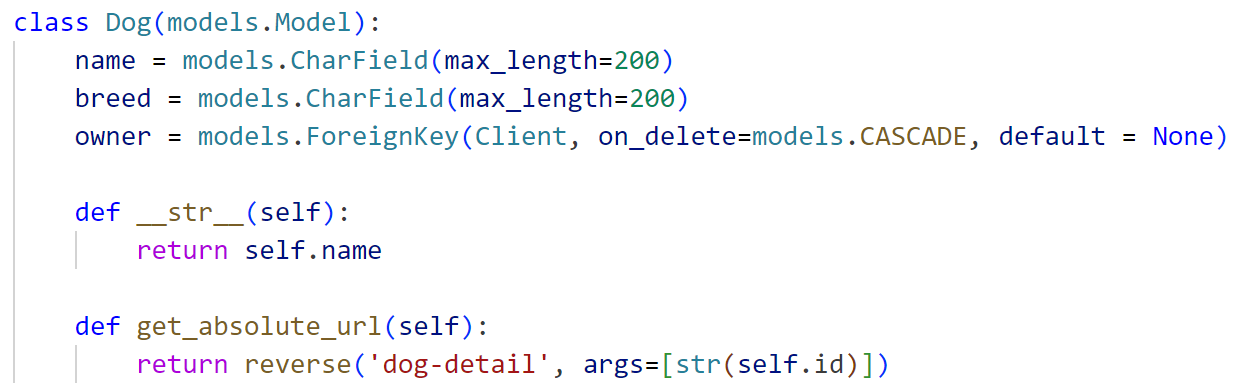
URL Namespace: ‘client-detail’

What to implement:

* Better List View that will show the profile picture of the user.
* More optional categories, such as phone number, description, etc.

**Model 2 – The Dogs**

Dogs belong to clients. The relationship is one to many, as one client can have as many dogs as they wish. This means that a dog can have one owner, but a client can have many dogs.



They have these associated views:

**DogListView**

Template: dog\_list.html

URL Namespace: ‘dogs’

**DogDetailView**

Template: dog\_detail.html

URL Namespace: ‘dog-detail’

**createDog**

Template: dog\_form.html

URL Namespace: ‘create-dog’

**updateDog**

Template: dog\_form.html

URL Namespace: ‘update-dog’

**deleteDog**

Template: dog\_delete.html

URL Namespace: ‘delete-dog’

What to implement:

* More optional categories, such as age, weight, height, medical, etc.

**Model 3 – The Dog Walker**

* *To be implemented in the next iteration –*

**Other Templates & URLs**

base\_template.html (Gives the navigation bar and is extended by all other templates, has the logo and the options to visit client and dog walker lists.

index.html (Homepage of the app, acts as a launching pad to the rest.)

**Folder Hierarchy**

fido\_hub – just like ‘django-project’ in the portfolio project hierarchy, where important project related code was added in settings.py.

fidohup\_app – the actual application, where the models, the templates, and the URLs are all placed. This is the majority of the code for the app to run.

myvenv – just like ‘djvenv’ in the portfolio project hierarchy, is home to virtual environment assets.

static – is home to the images folder.

.gitattributes – created when making the repository.

.gitignore – tells Git what to ignore when committing.

db.sqlite3 – the actual database.

manage.py – python file that can manage the virtual environment (stuff like creating super users, running the server, migrating the database, etc.)

requirements.txt – a list of all software and programs along with their versions used to make this application possible.

server commands.txt – a list of commands that is easy to copy and paste in order to run the server.