Garage Automation: Technical Documentation

Software Engineering – 14:332:452

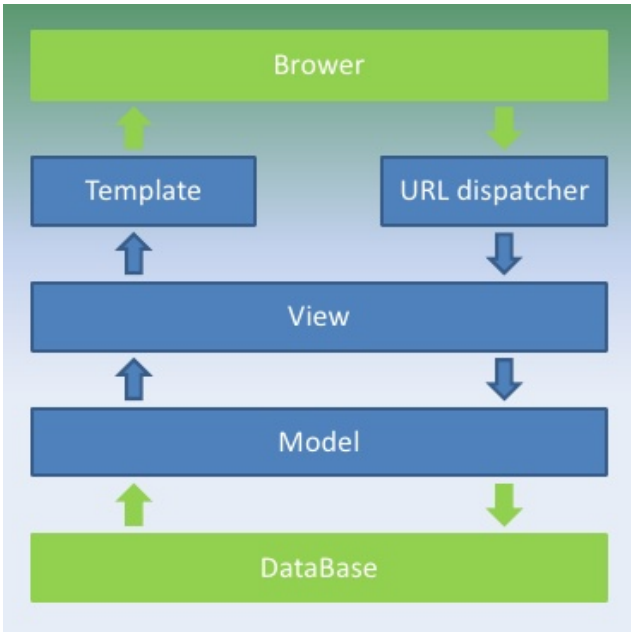
Group #10: Kendric Postrero, Gao Pan, Vatsal Pandya, Yunqi Shen, Arthur Rafal, Guy Rubinstein

URL:
https://vatsal09.github.io/Garage-Automation/

Submission Date:
May 1, 2017

The crux of the system guts and its execution happens through the Django framework. Django framework utilizes the Model View Controller software design pattern for developing web applications. MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. Django utilizes a variation of the MVC because Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request:



Due to the expansive nature of the code base, it would be difficult to explain each file. However, we explain the general characterization of each file and its purpose.

Django Root Folder (mysite/mysite folder)

The Django root directory will be named according to the project name you specified in django-admin startproject [projectname]. This directory is the project's connection with Django.

[projectname]/settings/

Instead of a plain settings-file, the configuration is split into several files in this Python module

[projectname]/urls.py
The root URL configuration of the project. The only configured set of urls is the admin-application. This set of urls is extended by the urls.py from each Django app.

apps/ (garageAutomation, Parking, and Manager)

This directory is used for custom applications. You can safely remove this directory, if you do not plan to develop custom applications. Most of a Django project's apps will be installed into the Python path and not be kept in your project root.

static/ and templates/

These directories are used for project wide files, meaning project wide static assets and templates. This includes the HTML, CSS, JS, background images, and etc. These are considered to be static assets by Django. You can split HTML blocks because the Django template engine enables the use of extending HTMLs.

Django Models (mysite/[app name]/models)

Django has an important functionality of using models. A model is the single, definitive source of information about our data. It is a class that represents a table or collection in our database, where every attribute of the class is a field of the table or collection. Models make it easy to define the structure of stored data because you can think of models as objects that can be instantiated if needed for functionality. We can choose in Django's settings file what database we want to use and we don't even need to talk to it directly. Using Models, Django handles all the dirty work of communicating with the database for you. Some important models that we had to create for our parking application is: Parking Lot and Session located in mysite/parking/

Parking Lot Model:
It has attributes: user, address, max level, and max spots. So in the database these attributes would be fields. User (in this case, the user is a manager) is a connected through a foreign key meaning there is a many to one relationship between user and parking lot. A manager can create and manage multiple parking lots. A parking lot is defined by its address while max level and max spots are just properties to help initialize a parking lot.

Session Model:
A session model defines a parking session in a specific parking lot. This model helps keep track of the current cars that have entered and exited the parking lot. It has attributes: parkingLot, Credit_Card, license_plate_number, time_arrived, time_exited, stay_length, user_type, and amount_charged. Parking Lot is defined with a foreign key meaning that a parking lot can hold many sessions. It generally holds information like license plate number, credit card information, and user type for later use in our views.py functions to execute the logic in our application. Time Arrived and time exited is always important to know for data collection purposes.

ActiveSession Model:
An ActiveSession model is a subclass of a Session model in the way that it inherits all of its fields. It is its own model, however, there is an automatic relationship that is created between a child and its parent. This relationship was chosen due to the similarities of the fields between the two models. An instance of each model is created when a customer enters the lot. The difference is that when the customer exits, the session instance saves the exit data while the ActiveSession instance gets deleted from the table.

Image Model:
An Image model consists of an ImageField and a DateTimeField. This model facilitates the uploading and storage of the actual image.

Django Views (mysite/[app name]/views.py)

Django puts all the “logic” of our application into a views.py file. Within the views.py are view functions that take a web request and returns a Webresponse. The views.py will request information about any model we created before and pass it to a template. This response can be the HTML contents of a web page, or a redirect, or a 404 error. Each view function contains whatever log is necessary to return that response. You can view our views.py files by checking out mysite/[app name]/views.py. In there each function is carefully commented and documented to define their key functionality. Since we integrated OpenALPR Cloud API, an API that analyzed images of a car’s license plate, we needed to import that library into our views.py file to use the API functions.