Provided are learning resources for our project. All of us are free to contribute

**Web Development:**

**Software Development Frameworks:**

MVC (Model, View, Control)

A general framework for developing software, very popular with web-applications

1. Model = The database, data for users, orders, supplies, etc. We can use SQL for this
2. View = front-end, how the program looks, the user interface. In web applications, its html, css, javascript, react, etc…

We also need to communicate with the backend, in web apps, its using AJAX

1. Control = backend logic: CRUD (Create, Read, Update, Delete) for databases, send content to the client (like HTML templates or objects), etc

earning resources for our project. All of us are free to contribute

MVT (Model, View, Template)

A more specific version of MVC. Django is based on this framework

1. Model = database model like in MVC
2. View = Same as controller in MVC except it must render HTML pages
3. Template: Same as view in MVC

**Web applications:**

HTTP

HTTP is a proto-call used to communicate the browser with the server (backend). Whenever you type a URL on a bowser, you are usually making an HTTP request to a server.

Browser -> request -> server

Server -> response -> Browser

URL

They are often used to render web pages but can also be used to send other types of content.

**Learn Django:**

Recommended:

<https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>

Video Tutorial

<https://www.youtube.com/watch?v=F5mRW0jo-U4&t=163s>

The link the video referred above to install django changed so this is the actual link: <https://www.codingforentrepreneurs.com/from-zero>

Another useful reference

<https://overiq.com/django-1-11/>

You can also refer to tutorialspoint but the code is rather depreciated:

<https://www.tutorialspoint.com/django/index.htm>

**Django database/SQL:**

This information will deal with how to design a database in Django. Keep in mind Django uses a library that corresponds easily with ER-diagrams

One\_to\_many relationship: <https://amir.rachum.com/blog/2013/06/15/a-case-for-a-onetomany-relationship-in-django/>

Many\_to\_Many relationship

<https://docs.djangoproject.com/en/2.2/topics/db/examples/many_to_many/>

**Web API**

When developing web apps, may need to get services from other servers over the internet. For example you may need it for email authentication, sending text messages, embedding google maps onto your app, etc. Web servers provide APIs specifically for that. You can also make your own API as well if you want. They typically send back non-html content where they must be properly interpreted.

In order to know how to use one on a web page, you need to learn AJAX, which is an API used to connect to web apis and send HTTP requests to them. Axios is a useful library which uses AJAX in the background to make our lives easier.

Axios:

<https://medium.com/codingthesmartway-com-blog/getting-started-with-axios-166cb0035237>

You can also use a web api in the backend too for security reasons.

For the delivery person, to get directions, we will use the google maps API:

<https://github.com/googlemaps/google-maps-services-python>

**Django REST API**

A Rest API is where instead of using URL paths in django to render HTML pages, we use them to send other types of content like text, objects, arrays, etc. It is essentially your custom web API. You would want to do this if you want to separate the front-end and the back-end of the web app, especially if you are using a front-end framework to make html pages and design their url paths using React.js or even Angular instead of using templates and views in Django. REST APIs are also needed if you want to make a server work not only with web-sites but also with mobile applications. We can get away without using a REST API for this project but if you want an app to scale to other platforms, it is highly recommended to know how to make one so here is a link on using Django to make one.

<https://www.django-rest-framework.org/>