Summer Internship

Report and Documentation

Backend Server
&
Simulator

Submitted by

Kartik Saxena
Final Year, CSE Department
National Institute of Technology Hamirpur

Under the guidance of

Cohan Sujay Carlos
C.E.O, Mechanical Chef

Mechanical Chef
65D, 1st Floor, Millers Rd, Cantonment Railway Quarters, Shivaji Nagar,
Bengaluru, Karnataka 560045
Summer Internship 2019

Abstract

Mechanical Chef is a mechanical and electronic marvel that can cook any Indian cuisine replicating the divine taste of Indian homemade food. It translates an algorithm of encoded steps for a recipe into mechanical movements and prepares the dish with utmost precision.

This project aims to build a back-end server, database design and basic user interface for the new Mechanical Chef website to control and monitor the mechanical chef. The website provides a functionality to register a new user, place orders on the mechanical chef, register a new robot, view the real time updates on each step of the recipe and get live video stream on the recipe being cooked.

This website encapsulates the overall user experience and automates the whole procedure from loading vegetables and spices in the dispensers to finishing a complete dish

Contents

Chapter 1

Objectives

1.1 Website and User Interface

The Mechanical Chef website is built primarily to automate the procedure of controlling the mechanical chef robot to cook a dish and monitor the real time cooking procedure through live video feed.

One of the foremost aim of this project is to build one such user interface for a quick and easy interaction between the user and the machine. There should be a functionality to register a user, maintain their profile, keeping a track of the dishes they have cooked as well as the robots they own.

The user should also be able to customize the recipes and record new recipes to be stored in the database.

1.2 Reliable Back-End Server

A robust back-end was needed for the website to handle the API requests as well as polling by the simulator. The back-end had to be secure and restricting unauthorized access to confidential data of the users as well as the robots.

The database design for the project had to be developed from scratch and preferably in any SQL supporting RDBMS for the comprehensibility of the developer, flexibility in design and scalability for any future work.

The framework for building the back-end was aimed to support rapid development as a multifold of modules had to be built and developed from square one. The back-end had to be versatile for building a secure user authentication system.

1.3 RESTful Application Program Interface (APIs)

To support the back-end design and encompass the overall user experience miscellaneous APIs had to be developed for user authentication, placing orders, fetching recipes being cooked etc.

Some APIs were needed by the simulator to perform the operations replicating the robots actions while cooking any dish. All these APIs were integrated with the framework used to develop the back-end server.

1.4 Poller

To establish communication between the simulator and the server, polling was required. A separate module (poller) was made to perform the same and was integrated with the simulator.

Polling can be done without the knowledge of IP address of the client and is thus more flexible than using sockets between a client and a server.

1.5 Simulator

Since the basic server could not be tested on the actual machine initially a simulator was built to test and debug the website and back-end server. The simulator module was meant to be run on each robot acting as an interface between the server and the robot itself.

Simulator for each machine had to be run on separate threads so that they don't interfere with each other while fetching commands from the server and each simulator had sub-threads for each vessel.

Chapter 2

Introduction

2.1 What we planned to build?

This project consists of three compound modules that can be further broken down into elementary sub-modules. These three modules are as follows :

- User Interface : For interaction with the user i.e. taking inputs and displaying outputs

- Backend Server : To handle API requests and connect database to the website

- Simulator : To simulate the working of a mechanical chef

2.2 User Interface

The interface used in this project is built using basic HTML, CSS, Bootstrap and JavaScript along with glyphicon icons for increased interactivity. It was meant to be minimalistic for the initial prototype.

The website as such is dynamic in nature and can adapt to interfaces like laptops, tablets and smart phones. The frontend work is integrated within the backend framework as templates.

These templates are HTML files which also include Jinja2 for executing python script inside the HTML code. Basic layout for the templates is chosen initially and layered over the other templates.

2.3   Initial assessment for server framework

Python was the initial choice for developing the backend and as such there were a few choices, these choices were narrowed down to Flask and Django as they are much more versatile and easy to work with. Django was the winner because of the following reasons :

- Django is a full-stack Python web framework, whereas Flask is a simple, lightweight, and minimalist web framework.

- Django makes it easier for users to handle common project administration tasks by providing a ready-to-use admin framework.

- Django comes with a built-in template engine that enables developers to define a web applications user facing layer without putting extra time and effort.

- Django allows developers to divide a project into multiple applications.

- Django comes with a built-in bootstrapping tool django-admin

2.4   Installation and Integration

Django comes with an initial admin framework along with priorly installed libraries for user registration and authentication.

Prerequisites for installing django :

- Python3 : sudo apt-get install python3

- Pip3 : sudo apt-get install -y python3-pip

- XAMPP : wget https://www.apachefriends.org/xampp-files/7.2.2/xampp-linux-x64-7.2.2-0-installer.run & chmod +x xampp-linux-x64-5.6.33-0-installer.run & ./xampp-linux-x64-7.2.2-0-installer.run

- To start XAMPP service : ./xampp-linux-x64-7.2.2-0-installer.run

- Installing virtual environment : pip3 install virtualenv

- Make directory for Django Application and open terminal in this folder.

- Setting up virtual environment : virtualenv env_name

- Activate the virtual environment : env_name/bin/activate

For installing django and starting a new project the following steps are to be followed :

- Installing Django : pip install django

- Starting a project : django-admin startproject ¡project name¿

- Running Server : python manage.py runserver IP_Address:Port

- Making a superuser : python manage.py createsuperuser

The basic Django libraries are not sufficient for developing a full fledged website. The following applications were installed using pip3 for the effective functioning of the website :

- crispy_report : For generating automatic forms from models

- phonenumber_field : To store phone numbers of the users

- django_mysql : For using mysql inside django

- django_unixdatetimefield : For using Date Time Field inside django models

2.5   Database Design and APIs

The database is built in MySQL RDBMS and hosted locally using Php-MyAdmin, the basic models of the MySQL database are supported by the django admin app and can be migrated into the database using the following command : python manage.py migrate

Moreover all the new models, built/updated inside the django framework can be logged using the command : python manage.py makemigrations and then can be migrated using : python manage.py migrate

The models built in this project are as follows :

- Profile : For storing the information of the user

- Robot : For storing the information of the robot

- Dishes : For storing the orders placed by the user

- Recipe : For storing the information of the recipe

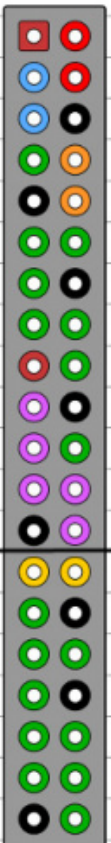- RobotUser : For storing link between user and robot

- UserRecipe : For storing customized recipes

- Simulator : For storing the commands for the simulator

The APIs built for accessing information from the user interface are :

- / : Fetch homepage

- /user : Fetch list of all users

- /user/id : Fetch details of particular user

- /user/signup : Register a new user

- /user/login : Login a user

- /user/logout : Logout a user

- /recipe : fetch list of all recipes and also to place order

- /recipe/id : fetch details of a particular recipe

- /robot : fetch list of all robots

- /robot/id : fetch details of a particular robot

- /robot/vessels/id : fetch the number of vessels of a particular robot

- /robotuser/user_id : fetch list of all robots owned by a particular user

- /robotuser/user_id/robot_id : fetch details of particular robot owed by a user

- /status/user_id : Fetch all orders placed by a user

- /status/user_id/dish_id : Fetch details of a particular order placed by a user

- status/user_id/order_id/robot_id/loading : Fetch the ingredients loading page for a particular order

- api/robot_id/fetch : Fetch initial instructions by the simulator when orders are unknown

- api/robot_id/user_id/order_id/fetch : Fetch initial instructions by the simulator when all orders and vessels are known

- camera : Fetch camera feed from the raspicam and display it on the website

2.6 Raspberry Pi Introduction

The RaspberryPi Zero can be used as a USB device or "USB gadget", plugged into another computer via a USB port on another machine. It can be configured in multiple ways. It is a micro-controller chip capable of performing various tasks and can control varied modules performing different functionalities.

Raspberry Pi2 GPIO Header

Pin#	NAME			NAME	Pin#
01	3.3v DC Power			DC Power 5v	02
03	GPIO02 (SDA1 , I²C)			DC Power 5v	04
05	GPIO03 (SCL1 , I²C)			Ground	06
07	GPIO04 (GPIO_GCLK)			(TXD0) GPIO14	08
09	Ground			(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)			(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)			Ground	14
15	GPIO22 (GPIO_GEN3)			(GPIO_GEN4) GPIO23	16
17	3.3v DC Power			(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)			Ground	20
21	GPIO09 (SPI_MISO)			(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)			(SPI_CE0_N) GPIO08	24
25	Ground			(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)			(I²C ID EEPROM) ID_SC	28
29	GPIO05			Ground	30
31	GPIO06			GPIO12	32
33	GPIO13			Ground	34
35	GPIO19			GPIO16	36
37	GPIO26			GPIO20	38
39	Ground			GPIO21	40

Early Models

Late Models

Rev. 1
26/01/2014   http://www.element14.com

In this project we have used the raspberry pi as a camera for taking snapshots from the machine and sending the stream of jpeg images to the website for live monitoring.

2.7 Polling Code

Polling is a term used to describe a continuous act of the computer or electronic device checking to verify connections on the computer or device are connected or operating properly.

In this project polling was done to fetch the instructions from the simulator model of the database and update the simulator states based on these inputs. The polling is performed once in every 5 secs for updating the states of the simulator running on different threads for each robot as well as for each vessel.

Although python has an inbuilt library that supports polling but here we have developed a poller module from scratch just to ensure flexibility of use and application.

2.8 Video Streaming Technology

For video streaming we have made use of Raspi Camera V2 for quick and easy use along with the Raspberry Pi Zero. HTTP protocol was used for transferring snapshots to the server in JPEG format. This stream of JPEG images was loaded directly into the website for live streaming.
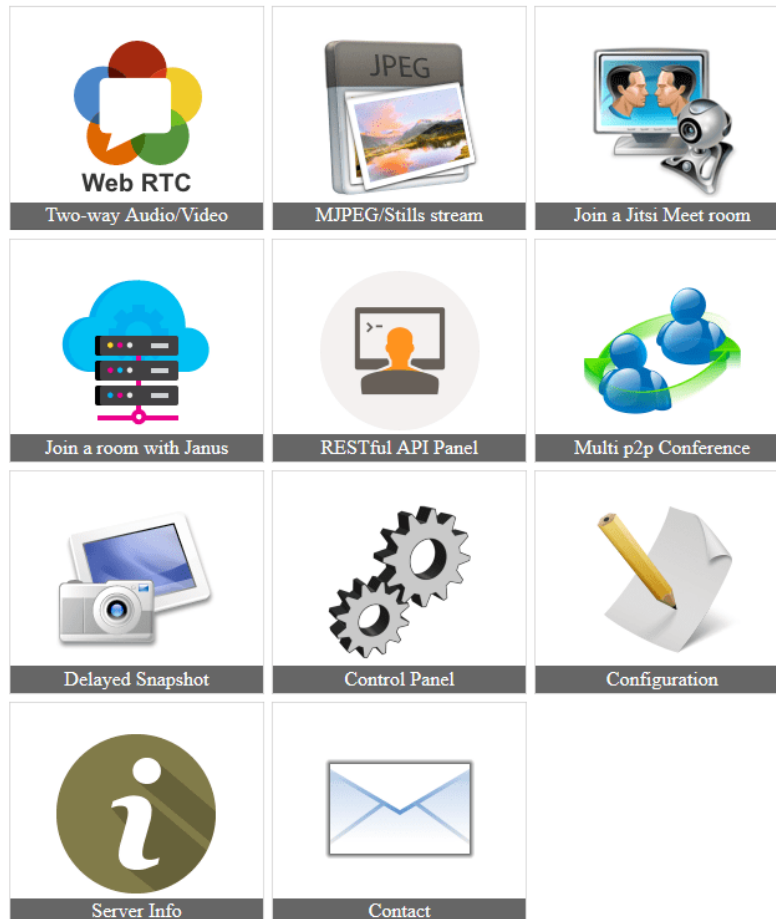
Although we also worked on UV4L streaming server to use MJPEG protocol to embed live video feed on the website, it could not be integrated in the final simulator through polling only.

2.9 Raspberry Pi Camera and UV4L Streaming server

The Raspi Camera V2 can be connected to the raspberry pi and has the functionality to shoot, record and take snapshots. With Open CV it can also be used to detect motion sensing and face detection.

The UV4L provides the following functionalities :

<u>UV4L</u> Streaming Server

Two-way Audio/Video	MJPEG/Stills stream	Join a Jitsi Meet room
Join a room with Janus	RESTful API Panel	Multi p2p Conference
Delayed Snapshot	Control Panel	Configuration
Server Info	Contact	

2.10   Tech Stack

- Frameworks : Django, Rasbian

- Languages : Python, HTML, CSS and Bootstrap

- Backend database : MySQL

- Hardware : Raspberry Pi Zero, Raspicam

Chapter 3

Work Done

3.1 User Interface

3.1.1 Initial Assessment

A basic user interface had to be developed for the website with all the functionalities that are to be provided to the user as well as the admin. Initial assessment of the user interface proposed the following functionalities :

- Login/SignUp : for registering the user

- Profile : for maintaining the profile of the user and to keep a record of the dishes prepared, robots used, recipes generated

- Profile : for maintaining the profile of the user and to keep a record of the dishes prepared, robots used, recipes generated

- Recipe Status : for giving a real time update on how the dish is being cooked and what all steps are being taken.

- Record : for giving step by step atomic instructions to the mechanical chef

3.1.2 Final Module

The final UI was developed using HTML, CSS, Bootstrap and JavaScript to keep a simple user friendly design for controlling the robot using the website.

Although this final module is built using primitive tech stack, the efficiency of the design is notable as it is able to handle multiple simulator polling along with basic requests from the users.

The HTML templates form up the templates part of the MVT (Model—View—Template) architecture of the django framework. These templates are stored inside the django project folder and are rendered using the views.py file corresponding to the the requested API.

3.2 Backend and Server

3.2.1 Initial Assessment

As already mentioned, the backend was to be developed in Django for its unique and developer friendly characteristics. It is very versatile and supports rapid application development.

The initial database that looked like this :
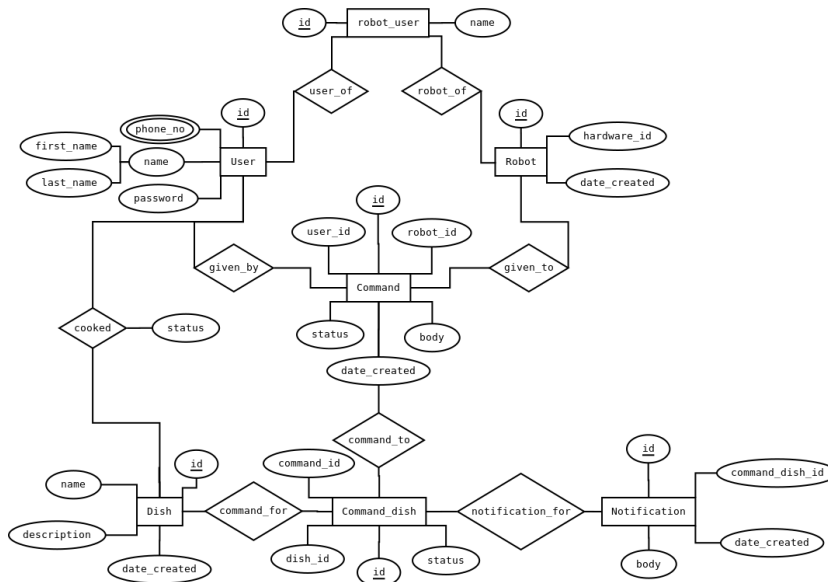
We implemented the following changes to the database :

- Clubbing together Command, Command_dish and Dish to make a Recipe table

- Dishes will be a separate table for storing the dishes that have been cooked by a person

- Adding a Login/SignUp functionality

The biggest question that arose initially was to decide where to host the server? Initially the server was hosted on the machine itself but we thought of another plan to host the server.

3.2.2 Detailed Description

We had three choices for hosting the server :

- Solution #1 : Hosting the web server on Raspberry Pi itself that is accessible to internet.

- Requirements :

    - Internet connection on the Raspberry Pi

    - An operating system on the Raspberry Pi (like Raspbian)

    - SSH access on Pi for remote access

    - Need a domain name for linking with the Pis address

- Technique : Port Forwarding to the Raspberry Pi at port not 80
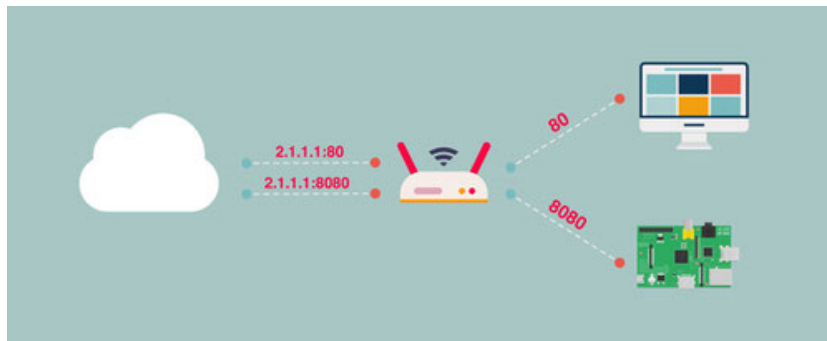
- Drawbacks

    - Raspberry Pi is still not as powerful as standard home PC.

    - Crash with heavy requests.

    - Wont be able to access the net because of all the traffic.

    - ISPs wont allow hosting a commercial server.

- Solution #2 : Hosting a web server on Raspberry Pi itself that can be accessed by any device on the same network i.e. on localhost

- Requirements :

    - Install dependencies required, mentioned in the article

    - Run Django server on local host

    - Edit the localhost address to Raspberry Pi IP

    - Access Raspberry Pi through any device on the same network

- Technique : Run the Django server locally on Pi

- Drawbacks

  - Cant access MechChef through net but can be operated very easily using any device on local network
  - Will fail to achieve the required remote access

- Solution #3 : To communicate with server hosted online with Pi as a client

- Requirements :

  - Internet connection on the Raspberry Pi
  - An operating system on the Raspberry Pi (like Raspbian)
  - SSH access on Pi for remote access
  - Send excel file to putty for easy access

- Technique : Use http protocols to access the online server and making requests (through polling)

- Drawbacks

  - Can only access mechchef through net
  - Cannot create local network for quick use

3.2.3 Final Module

The third solution was the most flexible solution and was incorporated into the final design of the server communication with the raspberry pi. This also ensured the independence of communication between server and machine without knowing the IP address of the machine or punching holes into the firewall of any wifi router.

3.3 APIs Built

3.3.1 Initial Assessment

The API that were thought of initially were based on the database architecture that was already being used in the previous website, they lacked a lot of important details and were reformed after a brainstorming session.

3.3.2 Final Module

The final APIs planned were as follows :

- / : GET

- /about : GET

- /user : GET

- /user/id : GET PATCH DELETE

- /user/signup : POST

- /user/login : POST

- /user/logout : POST

- /recipe : GET POST

- /recipe/id : GET PATCH DELETE

- /robot : GET

- /robot/id : GET

- /robot/vessels/id : GET

- /robotuser/user_id : GET POST

- /robotuser/user_id/robot_id : GET PATCH DELETE

- /status/user_id : GET POST

- /status/user_id/order_id : GET PATCH DELETE

- /status/user_id/order_id/robot_id/loading : GET

- /api/robot_id/fetch : GET

- api/robot_id/user_id/order_id/fetch : GET

- /camera : GET POST

3.4   Polling and Simulator

3.4.1   Initial Assessment

For writing the initial simulator we used a basic python code for a very shabby state based program that did not differentiate between the models used, views, controller or observer.

The initial poller was developed using the predefined python library called polling. This library supported many functions for checking the status of polling but did not provide enough means to access the information fetched after polling so we had to build our very own polling module.

3.4.2   Detailed Description

We found a couple of ways to build the simulator, one of which included using an MVC design pattern to construct the architecture of the simulator.

After reading through the "Design Patterns Elements of Reusable Object-Oriented Software" book, we managed to build the Simulator using the MVC model along with an Observer, it consisted of five modules :

- Simulator Module : For simulating the state transitions with given inputs

- Controller Module : For fetching information from the Views module and relaying them to the Simulator module

- View Module : For interacting with the user and taking inputs

- Observer Module : For displaying the outputs from the Controller to any platform (maybe command line or web)

- Poller Module : It was a customized polling module that was used to make requests to the server and update the states based on inputs given by the user

3.4.3   Final Module

The final module was built as planned as it provides a structured way to view the control flow and was easier to test. The unittest library of python was used to perform unit testing of the simulator based on multiple test cases.

3.5 Video Streaming

3.5.1 Initial Assessment

The final part of the internship was to finish the video streaming module of the simulator to fetch a video stream from the machine and show live feed on the website as the recipe is being cooked.

3.5.2 Detailed Description

There existed three solutions for implementing the Video Streaming Module

- Solution#1: Embedding the mjpeg video feed inside an HTML tag within the views of the website.

- Pros and Cons :

  - It was the easiest way to video stream data from the raspicam to the website and was pretty efficient
  - It required the IP address of the Raspberry Pi thereby defeating the purpose of using polling altogether

- Solution#2: Using UV4L streaming server on raspberry pi to send video feed to the server.

- Pros and Cons :

  - It was pretty difficult to fetch the video stream from the UV4L server and re-stream the fetched stream from raspicam on the website
  - The functionalities provided by the UV4L server were very advanced and could be put to good use

- Solution#3: A very basic solution was to send snapshots from the raspicam to the website and displaying the snapshots in realtime on the website.

- Pros and Cons :

  - It was fairly easy to implement after learning about the static and media files in django developement
  - It did not require IP address in the Raspberry Pi
  - It was near realtime unlike the other two solutions which were actually real time video feed

3.5.3 Final Module

For implementing a cost effective and simple solution we used the third solution as it fulfilled the criteria set up through polling and could easily be integrated along with the poller module.

Chapter 4

Future Scope

4.1 User Interface

The user interface is built in basic HTML, CSS and JavaScript in this prototype, this has to be built using another library called ReactJS for much more user friendly designs as well as better interactivity.

ReactJS recently became popular after Facebook used it for their frontend. The special thing about React is that instead of updating entire DOM every time something in the state of website changes, it just updates the specific node which updates. This ensures faster and optimized user interaction. As Mechanical Chef soughts to stream live video and cooking stream, faster UI is a must have.

4.2 Record a recipe

The functionality for preparing a customized recipe and recording it in the database is a very complex problem yet to be solved. The recipes are encoded in three of the following steps for now :

- Heat

- Stir

- Drop

The procedure for preparing an algorithm from a recipe is a tedious task and requires a specialist to code all the steps into an Excel file which can be fed to the robot. This however is not possible for a user and thus a proper UI had to be developed for the same. Voice inputs are being considered a viable option.

4.3   Hosting a Local Network on the Robot

For users who do not have an access to the internet, a personal area network (maybe through bluetooth) can be hosted on the machine locally and can connect to the user for a quick preparation of dishes or for demonstration purposes.

For this personal area network a separate storage had to be embedded into the robot for making a public account and storing the user information on the server when it gets an access to the internet.

4.4   User Login through OTP

After a brainstorming session and weekly discussions we decided that the user registration process had to be kept much more simpler and quick for the comfort of the user.

This could be achieved by finding a faster way to register a user (even without internet access on a local network hosted on the machine) this could be made possible by registration using an OTP sent to the mobile number.

Chapter 5

Conclusion

The first prototype of the full-fledged mechanical chef website (written in python) is ready and supports all functionality from registering a user to placing orders to simulating the process of preparing a dish .

The simulators can easily communicate with the server and prepare a dish based on the inputs given by the user. All the processes are monitored through the django-admin as well as the central database administrator.

Except being a part of the website development project, we also worked on the actual robot, preparing and tasting recipes everyday, learning about the various components of the robot and catapulting it from version 2.0 to version 2.1 and then to version 3.0

Acknowledgment

This project was only possible with the esteemed guidance of my mentor Cohan Sujay Carlos, C.E.O, Mechanical Chef. His expertise and insight in python language was truly inspirational and has definitely transformed my coding style while building the project.

I would like to thank my colleagues from various technical backgrounds who have always tried their level best to solve my problems through brainstorming and expert advices.

I would also like to thank mechanical chef for giving me an opportunity to learn how startups actually work. The charisma and enthusiasm that I have experienced in this internship is unparalleled. I feel that through mechanical chef, I have gained an insight in food sciences and a love for cooking.

This internship wasn't just about building a website but also about exploring a lot about ideation, brainstorming, teamwork, planning, designing, implementation, testing and debugging while actually witnessing it.

References

[1] Django Crash Course - https://www.youtube.com/watch?v=D6esTdOLXh4

[2] Create Superuser Django - https://tutorial.djangogirls.org/en/django_admin/

[3] Polling - https://www.computerhope.com/jargon/p/polling.htm

[4] Python Polling Library - https://pypi.org/project/polling/

[5] Design Patterns : Elements of Reusable Object-Oriented Software

[6] Installing Django - https://www.digitalocean.com/community/tutorials/how-to-install-django-and-set-up-a-development-environment-on-ubuntu-16-04

[7] Django Exceptions - https://docs.djangoproject.com/en/2.2/ref/exceptions/

[8] UV4L Documentation and Streaming Server - https://www.linux-projects.org/documentation/

[9] Raspberry Pi Remote Access - https://www.raspberrypi.org/documentation/remote-access/access-over-Internet/

[10] Installing XAMPP - http://devopspy.com/linux/install-xampp-ubuntu-16-04-using-terminal/