Python Project

# Abstract

To develop and chart machine downtime over a specified time frame utilizing the skill matrix as a basis for the requirements.

# Skill Matrix

|  |  |
| --- | --- |
| **Skill** | **Location in Code** |
| Output text (print with formatting) | views.py lines 864 – 978 |
| Output formatted numeric data | views.py lines 864 – 978 |
| Simple variables (string, integer, float) | models.py  lines 122-143 |
| Complex Variables (objects, multi-part) | models.py  lines 116-120 |
| Conditionals (standard if/then/else, switch) | models.py  lines 122-143 |
| Loops (for loops and/or while loops, iterators) | views.py lines 864 – 978 |
| Array (single dimension) | views.py lines 864 – 978 |
| Multidimensional array or dataframe | views.py lines 864 – 978 |
| Function with input (takes input) | views.py lines 864 – 978 |
| Function with output (returns output) | views.py lines 864 – 978 |
| Data/Math Library (pandas, numpy) | models.py  lines 122-143 |
| Data Library (plotting) | charts.djhtml  lines 13-117 |

# Problem Statement

Tasked with creating a web module that can track machine status in a production shop. The module will include Status of the machines, how many times each machine goes down, and for how long. The reporting with be graph based with downtime in hours, and the number of times it went down. Statistical analysis will be done on the final numbers, extrapolating the mean down time, how many times all machines went down, finally giving a total loss on downtime. A special rule must be used to only calculate downtime excluding weekends. Below are the formulas:

# Methodology

Once the models are created for the application, the business logic for calculating the down time will utilize Numpy. Numpy has a function built in that will check to see if a date is a business day or not. By giving Numpy a date range we can extrapolate only the business weekday. Once we have the number of business days the machine is down, we then calculate the date time difference into hours. This allows for a quick and easy calculation. If a machine does not have a finished downtime report i.e.(not repaired) then that time is not calculated in the report.

When the user wants to run a report, they simply go to the predetermined date ranges and run the report. They are then sent to the URL with the report they wanted to run. A for loop allows the script to cycle through all the data needed for calculations. This will display chart data of the different machines that have downtime reports attached to them within that date range. The various charts available are Life, Current Year, with Current Year Q1-Q4.

The charts are created with a JavaScript library called Charts.js. This allows for extreme customization on the chart and displaying of the data. Basically, the chart object is comprised of an JSON API that handles all the data and options.

The information can allow upper management to see if a machine needs to be replaced before it becomes a serious problem. Currently we rely on our Machine Maintenance’s memory to see if a machine needs to be replaced. With concrete evidence all stake holders are held accountable for the health of the machine shop.

# Solution

Being a Django Application the scripts can’t run without all the files and settings. What I have done is shown the code and how the data is displayed on the live server. Currently the application is roughly 3,000 lines of code. All of the code will not be listed, just the main aspects that deal with the business logic. The rest can be found within the GitHub Repository the entire application is available minus data from the database.

[GitHub Repository](https://github.com/starwriter34/EGN3214/tree/master/Final%20Project/mmaint)

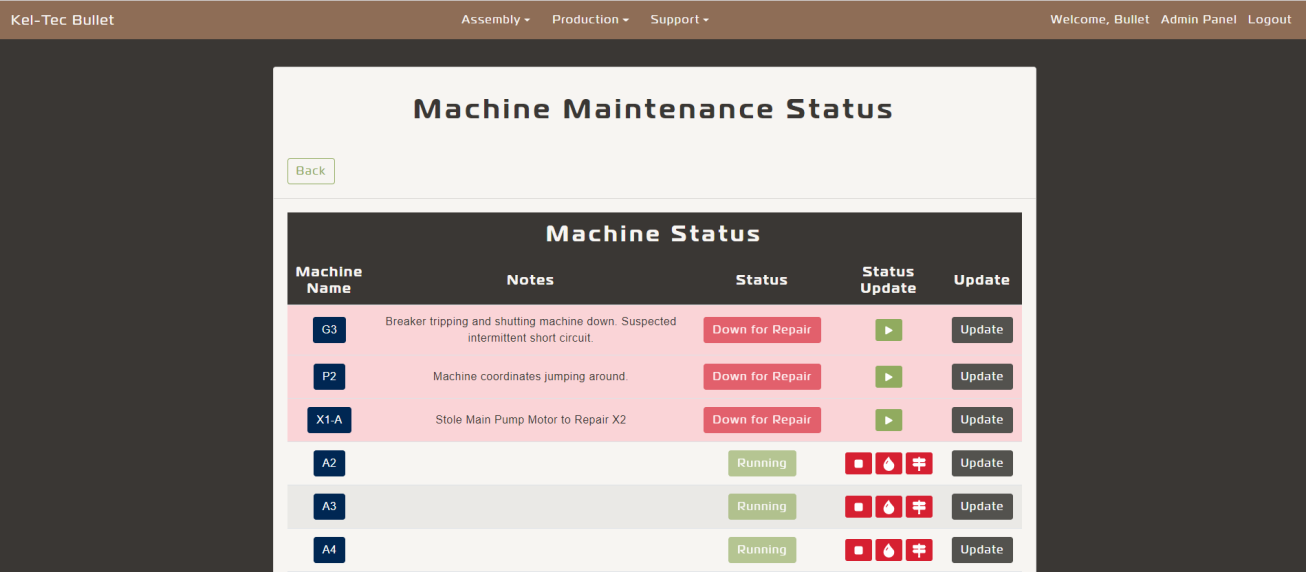


Figure : Machine Status View

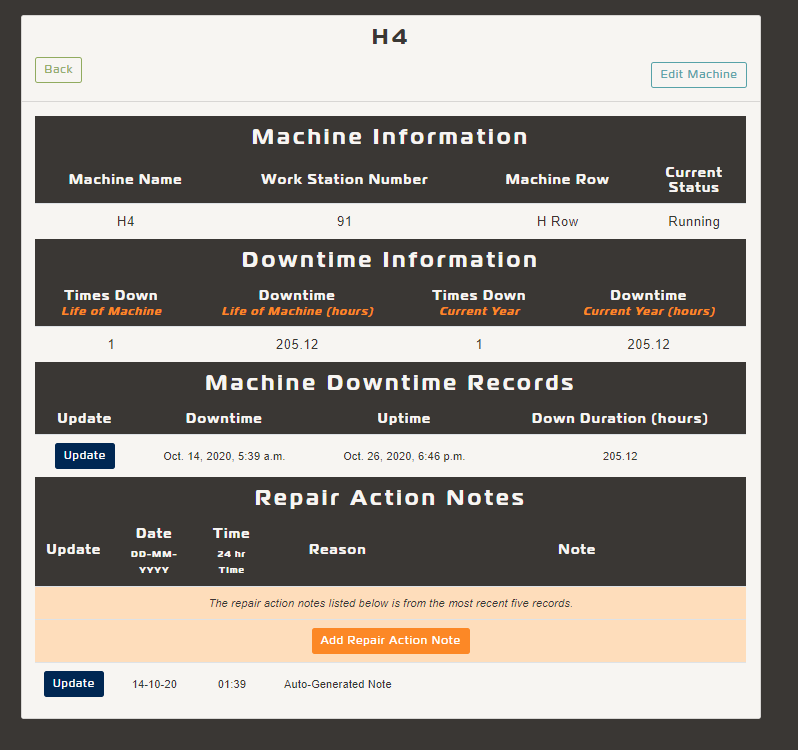


Figure : Machine Detail View

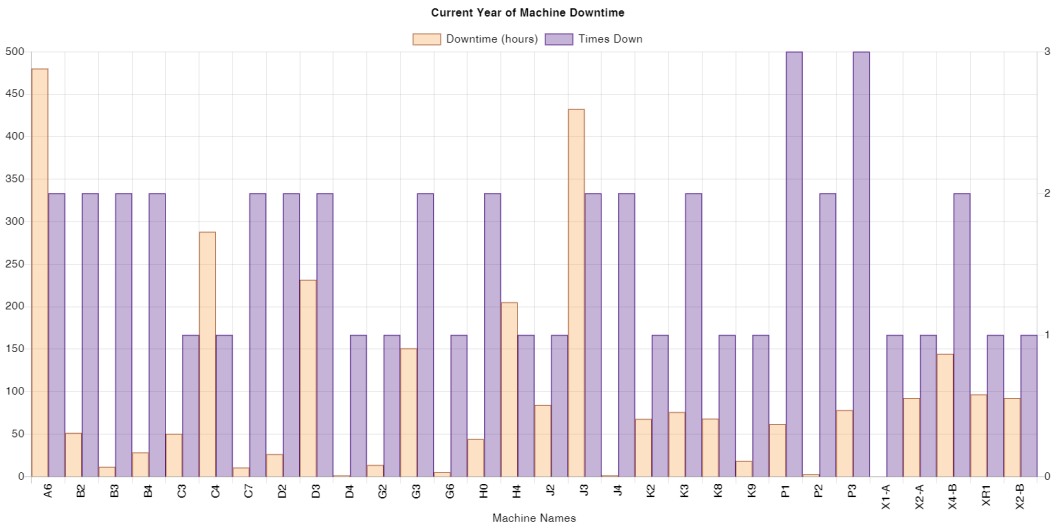


Figure : Chart View for Current Year

# Conclusion

The Machine Maintenance module is an ever evolving application that fills a need for the company. It has allowed management to make decisions to place the company in a better position. Finding a glaring problem and having the skills to implement a solution that can better a company you work for is an awesome feeling.

# Appendix

Code Snippets are from the listed files. To see all the code pertaining to the application please go to the GitHub Repository listed in the solution.

views.py

class MachineStatusCharts(TemplateView):

template\_name = 'mmaint/reports/charts.djhtml'

def get\_context\_data(self, \*args, \*\*kwargs):

chartType = self.kwargs['chartType'].lower()

average = []

downtime\_records = []

downtime\_records\_count = []

year = int(datetime.now().year)

context = super(MachineStatusCharts, self).get\_context\_data(\*\*kwargs)

context['names'] = MachineDownReports.objects.values\_list('machine\_id\_\_machine\_name').distinct()

if chartType == 'life':

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Life'

elif chartType == 'year':

start\_year = date(year,1,1)

end\_year = date(year,12,31)

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]) & Q(endtime\_\_gte=start\_year) & Q(endtime\_\_lte=end\_year))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Current Year'

elif chartType == 'yearq1':

start\_year = date(year,1,1)

end\_year = date(year,3,31)

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]) & Q(endtime\_\_gte=start\_year) & Q(endtime\_\_lte=end\_year))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Current Year First Quarter'

elif chartType == 'yearq2':

start\_year = date(year,4,1)

end\_year = date(year,6,30)

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]) & Q(endtime\_\_gte=start\_year) & Q(endtime\_\_lte=end\_year))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Current Year Second Quarter'

elif chartType == 'yearq3':

start\_year = date(year,7,1)

end\_year = date(year,9,30)

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]) & Q(endtime\_\_gte=start\_year) & Q(endtime\_\_lte=end\_year))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Current Year Third Quarter'

elif chartType == 'yearq4':

start\_year = date(year,10,1)

end\_year = date(year,12,31)

for record in context['names']:

total\_lists = MachineDownReports.objects.filter(Q(machine\_id\_\_machine\_name=record[0]) & Q(endtime\_\_gte=start\_year) & Q(endtime\_\_lte=end\_year))

totaldowntime = 0

downtime\_count = 0

for total in total\_lists:

if total.endtime == None:

totaldowntime += 0

else:

totaldowntime += float(total.total())

downtime\_count += 1

downtime\_records.append(round(totaldowntime,2))

downtime\_records\_count.append(downtime\_count)

context['chart'] = 'Current Year Fourth Quarter'

# Gather Data

averageLength = len(downtime\_records)

downtime\_average = round(sum(downtime\_records)/sum(downtime\_records\_count),2)

average.append(float(downtime\_average))

context['labels'] = list(context['names'])

context['average'] = average\*averageLength

context['downtime\_records'] = downtime\_records

context['downtime\_records\_count'] = downtime\_records\_count

return context

models.py

class MachineDownReports(models.Model):

machine\_id = models.ForeignKey('MachineStatus', on\_delete=models.CASCADE, verbose\_name='Machine Number')

starttime = models.DateTimeField(auto\_now=False, null=True, blank=True,)

endtime = models.DateTimeField(auto\_now=False, null=True, blank=True,)

status = models.CharField(max\_length=20, null=True, blank=True,)

def total(self):

if self.endtime == None and self.status == 'Machine Down':

busday = 'Machine Down No Calculation'

else:

if np.busday\_count(datetime.date(self.starttime), datetime.date(self.endtime)) > 1:

if datetime.time(self.starttime) == datetime.time(self.endtime):

busday = np.busday\_count(datetime.date(self.starttime), datetime.date(self.endtime))

busdayhours = self.endtime-self.starttime

busday = (busdayhours.seconds+(busday+1)\*(86400))/3600

busday = f'{busday:.2f}'

else:

busday = np.busday\_count(datetime.date(self.starttime), datetime.date(self.endtime))

busdayhours = self.endtime-self.starttime

busday = (busdayhours.seconds+(busday\*86400))/3600

busday = f'{busday:.2f}'

else:

busdaytd = self.endtime - self.starttime

busday = busdaytd.seconds / 3600

busday = f'{busday:.2f}'

return busday

class Meta:

verbose\_name = 'Machine Down Reports'

verbose\_name\_plural = 'Machine Down Reports'

def \_\_str\_\_(self):

return f'Machine {self.machine\_id}'

@classmethod

def machinedown\_re(self, pk):

MachineDownReports.objects.create(machine\_id\_id=pk, starttime=datetime.now(), status='Machine Down')

@classmethod

def machinerunning\_re(self, pk):

getrecord = MachineDownReports.objects.filter(machine\_id\_id=pk, endtime=None)

getrecord.update(endtime=datetime.now(), status='Machine Running')

charts.djhtml

{% extends "chart.html" %}

{% comment %} Title for Header on main page {% endcomment %}

{% block title%}

Machine Maintenance Charts

{% endblock title %}

{% comment %} Chart Date URL from urls.py {% endcomment %}

{% block chart\_url %}

{% endblock chart\_url %}

{% comment %} The script for the chartjs.org {% endcomment %}

{% block chart %}

<script>

var ctx = document.getElementById('chart');

Chart.defaults.global.defaultFontColor = 'black';

Chart.defaults.global.defaultFontSize = 16;

Chart.defaults.scale.gridLines.color = "rgba(127, 127, 127, 0.2)"

Chart.defaults.global.animation = 0

var myChart = new Chart(ctx, {

type: 'bar',

data: {

labels: [{% for label in labels %}'{{label.0}}',{% endfor %}],

datasets: [{

yAxisID: 'downtime',

label: 'Machine Downtime (hr)',

data: [{% for item in downtime\_records %}{{item}},{% endfor %}],

backgroundColor: 'rgba(249, 159, 64, 0.3)',

borderColor: 'rgb(249, 159, 64)',

borderWidth: 1,

categoryPercentage: 1.0,

barPercentage: 1.0,

hoverBackgroundColor: 'rgb(249, 159, 64)',

order: 2,

},

{

yAxisID: 'count',

label: 'Times Down (\*n)',

data: [{% for item in downtime\_records\_count %}{{item}},{% endfor %}],

backgroundColor: 'rgba(66, 9, 127, 0.3)',

borderColor: 'rgb(66, 9, 127)',

borderWidth: 1,

categoryPercentage: 1.0,

barPercentage: 1.0,

hoverBackgroundColor: 'rgb(66, 9, 127)',

order: 3,

},

{

label: 'Machine Downtime Overall Average (hr)',

data: [{% for item in average %}{{item}},{% endfor %}],

backgroundColor: 'rgba(96, 162, 169, 0.3)',

borderColor: 'rgb(96, 162, 169)',

borderWidth: 10,

pointRadius: 0,

hoverRadius: 0,

//pointStyle: 'rectRot',

fill: false,

type: 'line',

hoverBorderColor: 'rgb(96, 162, 169)',

hoverBackgroundColor:'rgb(96, 162, 169)',

steppedLine: 'after',

order: 1,

}],

},

options: {

responsive: true,

layout: {

padding: {

left: 10,

right: 10,

top: 10,

bottom: 10

}

},

title: {

display: true,

text: '{{chart}} for Machine Downtime' ,

},

tooltips: {

mode: 'index',

intersect: true

},

legend: {

display: true

},

scales: {

yAxes: [{

// Machine Hours Count

display: true,

position: 'left',

id: 'downtime',

},

// Machine Down Counts

{

display: true,

position: 'right',

id: 'count',

ticks:{

beginAtZero: true,

precision: 0,

},

gridLines: {

drawOnChartArea: false

}

}],

xAxes:[{

ticks: {

padding: 10,

}

}]

}

}

});

</script>

{% endblock chart %}