Criterion A

Defining the problem:

"*JB Topografia*" is a small business which works in the area of Topography. This business has recently been having issues with their profit. Their topography services are worked on by 4 contracted workers and 1 other worker called "Julio Baptista", which is the owner of the business. Originally, *JB* (Julio Baptista) operated as a freelance topographer. Due to a high income of requests for jobs, he saw himself obliged to contract 4 workers. Although all workers were functioning, a reduction in price per service was noticeably occurring, from 350€ per day, all the way to 200€ per day. This is due to the personalization and high quality of service in "*JB*" as a freelancer, which contracted workers were not skilled enough to succeed in. Thus, clients were not willing to pay past prices. *JB Topografia* has approached me in order to help with a business decision. *JB Topografia* has 3 main options as a solution, but they need help analyzing statistics and forecasts. The business agreed to meet me for further discussion on this topic, for me to create a website or software to assist.

Rationale

I will be creating a website for *JB Topografia*. This website will mainly consist of various business analyzing and decision-making tools. My product should, in its final form, assist *JB Topografia* in creating a solution to their problem, as well as helping other businesses in the future.

After conducting a meeting with the owner of *JB Topografia*, he explained to me in good detail what data he had acquired when it comes to forecasts and past market data. With this *JB* requested that I program at least 2 tools to analyze data, and 2 tools to assist in decision making. I was also told that the website should be extremely simple, and since no consumers will be visiting, it can be extremely bland. Since I will be utilizing only Python and HTML, this is great since it is not easy to have an advanced looking aesthetic without other languages. My website in general will have a main page where *JB Topografia* and other businesses will go into first, and then 4 different options for the 4 different tools. When entering one of the tools, input boxes will show up, with basic data requests. After submitting, the website will take you to the final product, consisting of a data, graph, or diagram.

The 2 decision making tools which I chose were a Decision Tree, and a Force Field Analysis. Both tools can be seen below:

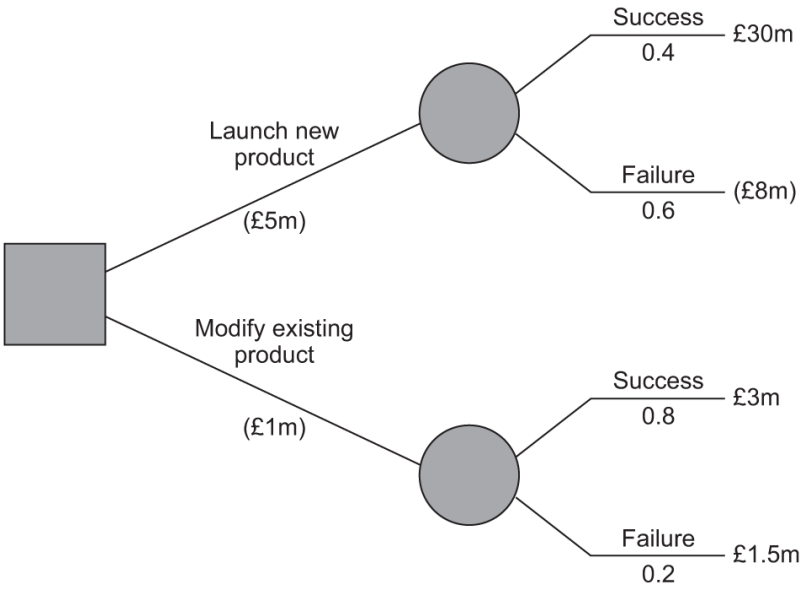


Image by aqa.org.uk

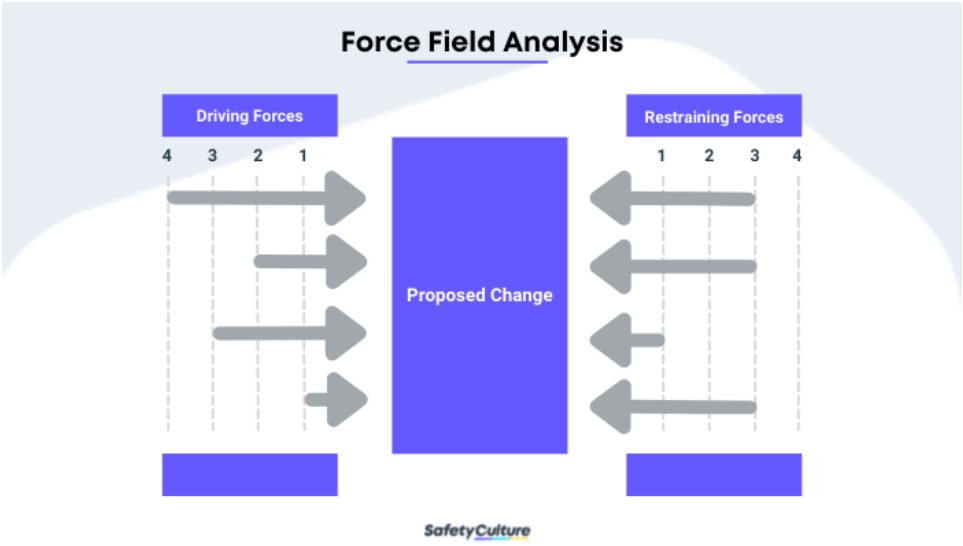


Image by safetyculture.com

The 2 business analyzing tools which I have decided to program is a break-even analysis graph, and a simple graphing tool. Break-even analysis graph can be observed under:

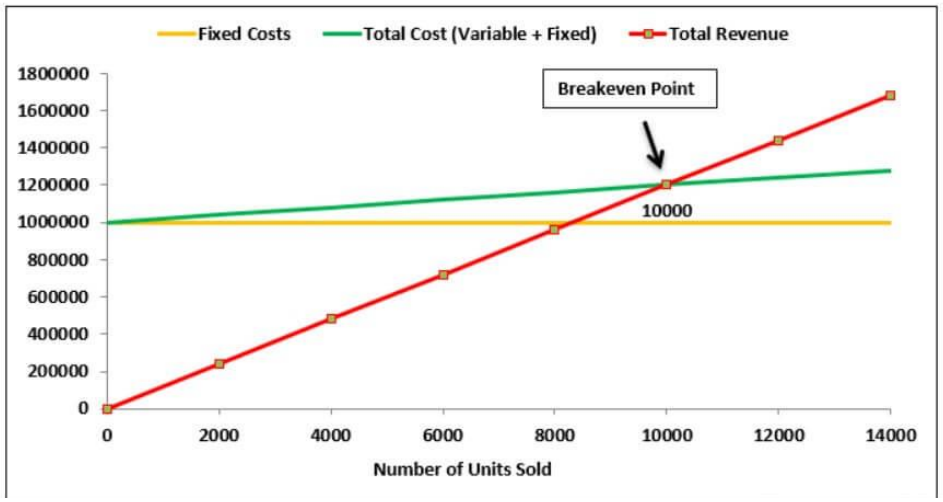


Image by wallstreetmojo.com

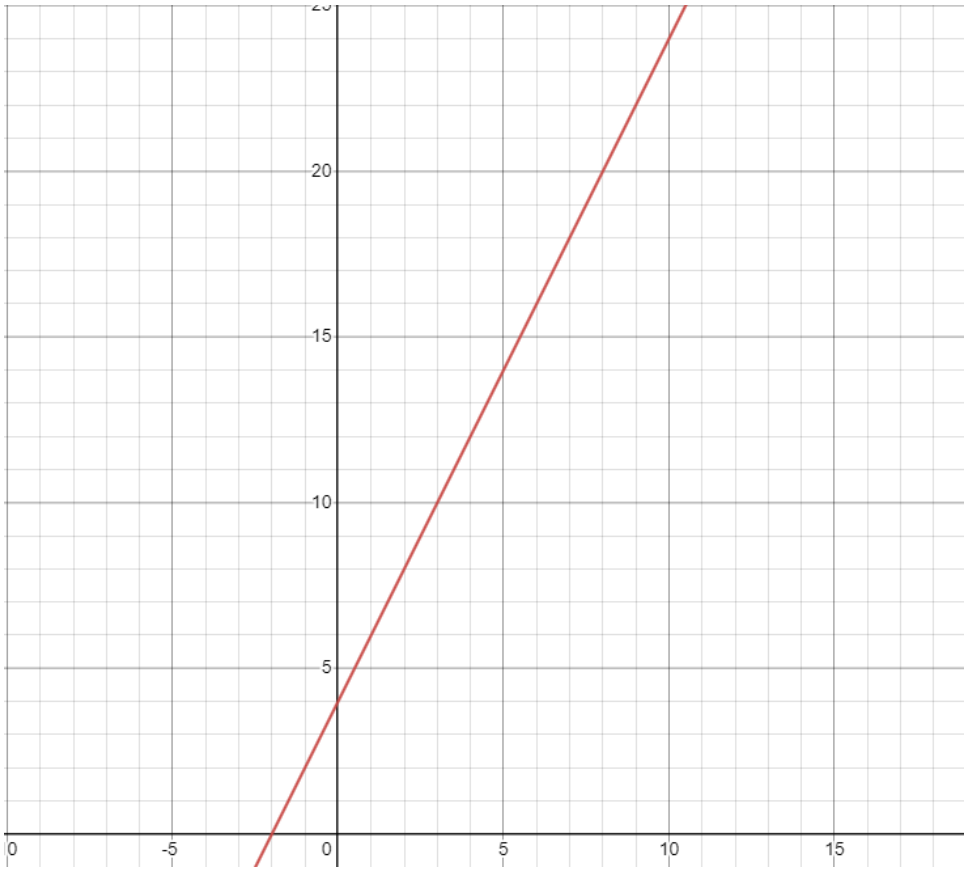


Image by desmos.com

For my website I will be utilizing the Django Framework. Django is a python-based web framework. I am using Django because Python is the coding language which I am the most comfortable with and Django is one of the most efficient and up-to-date web development frameworks in Python.

Criterion for Success

1. Create a working homepage for my website
2. Homepage will give access to all business tools
3. Entering a tool will give the user input boxes which will be added to the database when submitted.
4. Submitted data will be used to create business tools.
5. The following tools should function: Decision Tree, Graphing Tool, Force-Field Diagram, and Break-even Analysis.
6. Tools should be comprehensive and somewhat guided. (Labels, Comments, etc...)
7. After utilizing the tool the data should be removed from the database.
8. A working admin login.

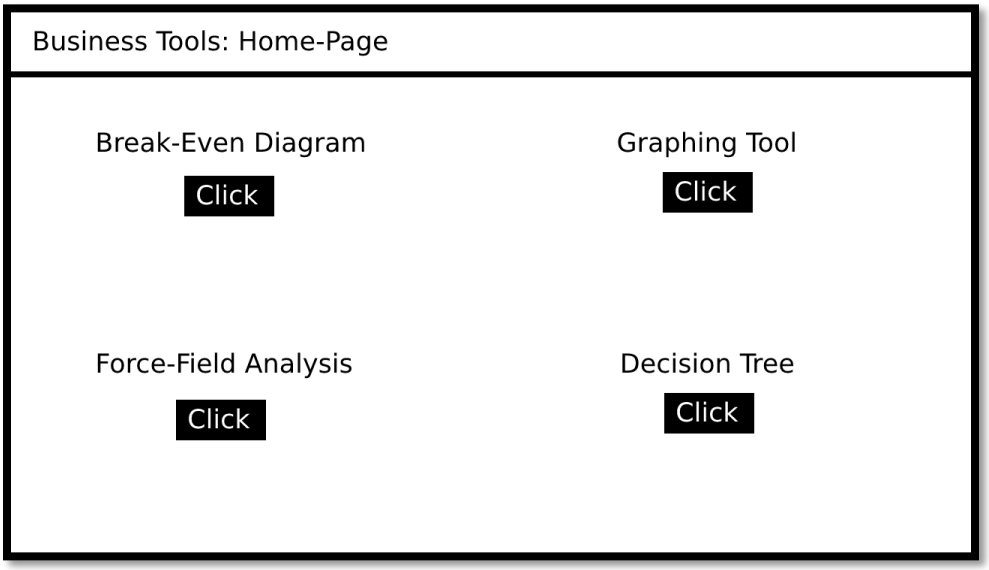
605 words

Criterion B

Diagram Examples – Proposed Design

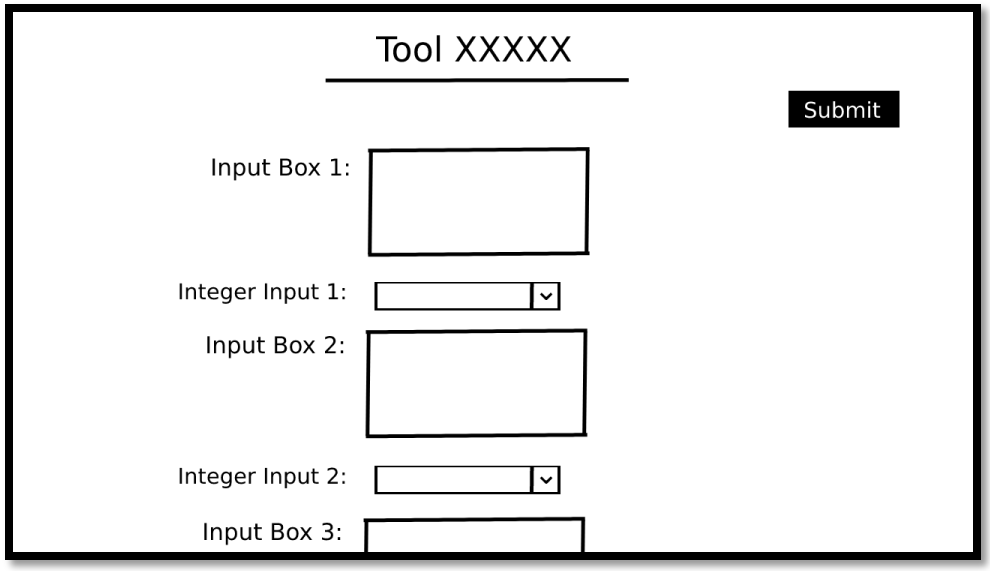
Main Page:

Figure 1: Main Page Diagram



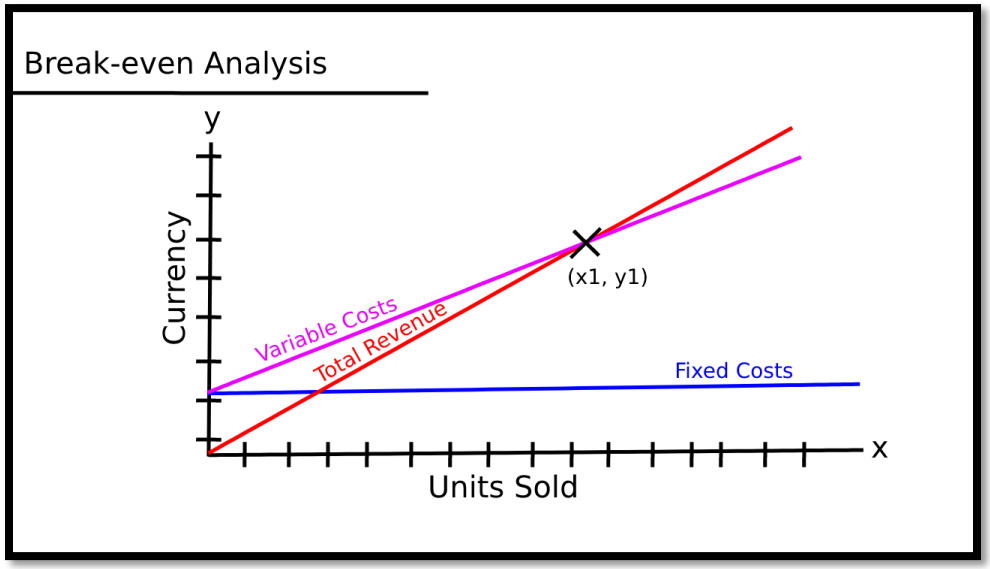
Input Pages:

Figure 2: Input Pages Diagram



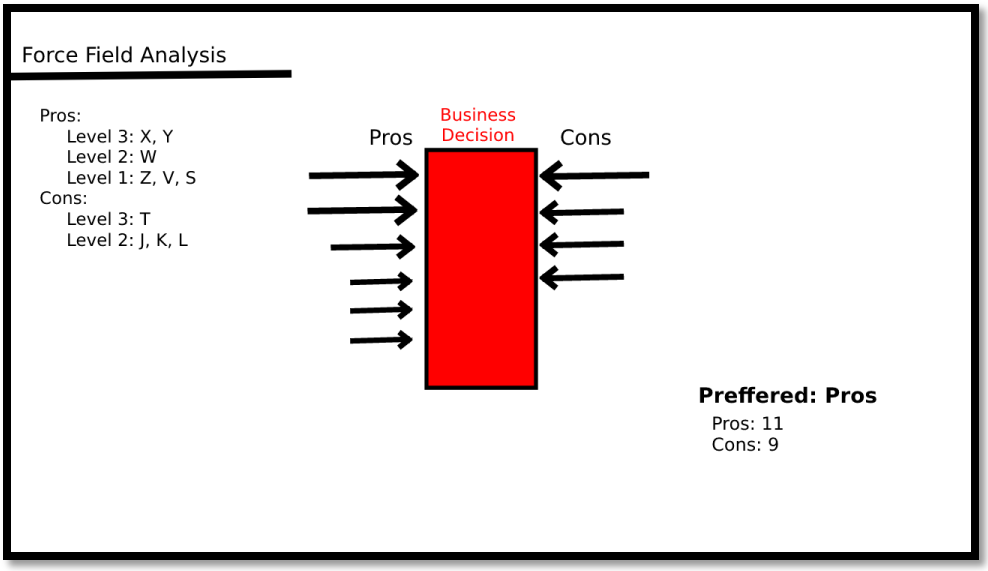
Break-even Analysis:

Figure 3: Break-even Diagram



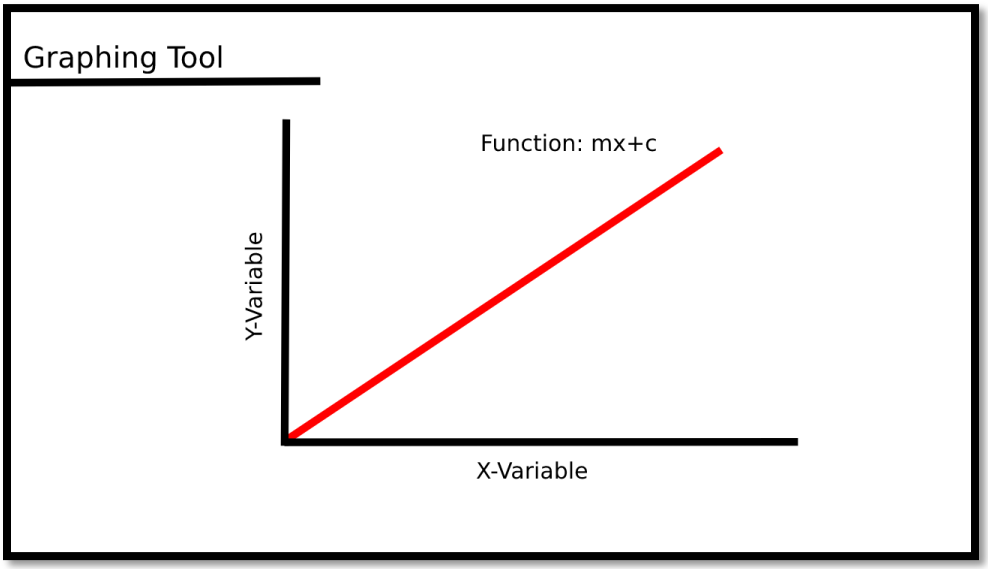
Force-Field Analysis:

Figure 4: Force-Field Diagram



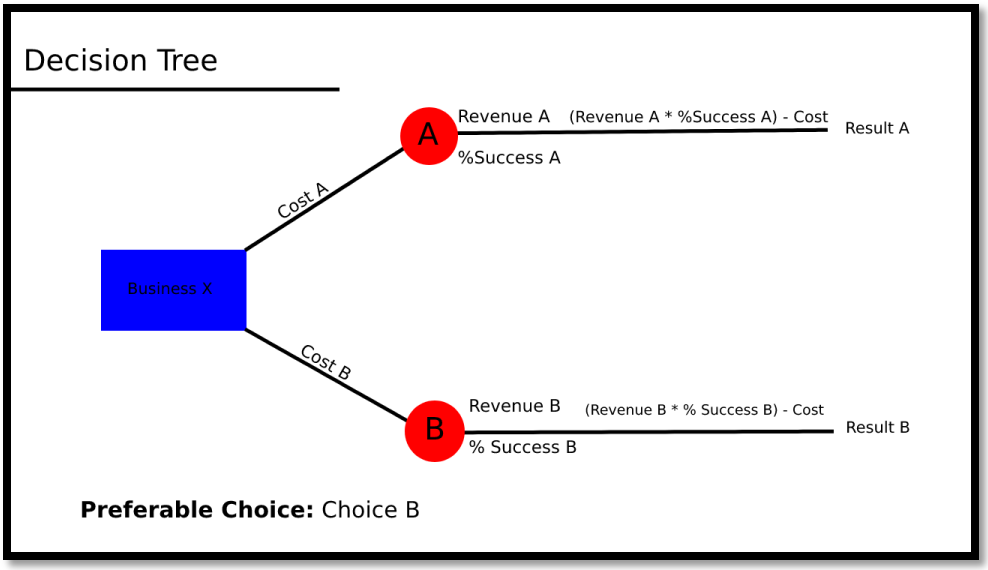
Graphing Tool:

Figure 5: Graphing Diagram



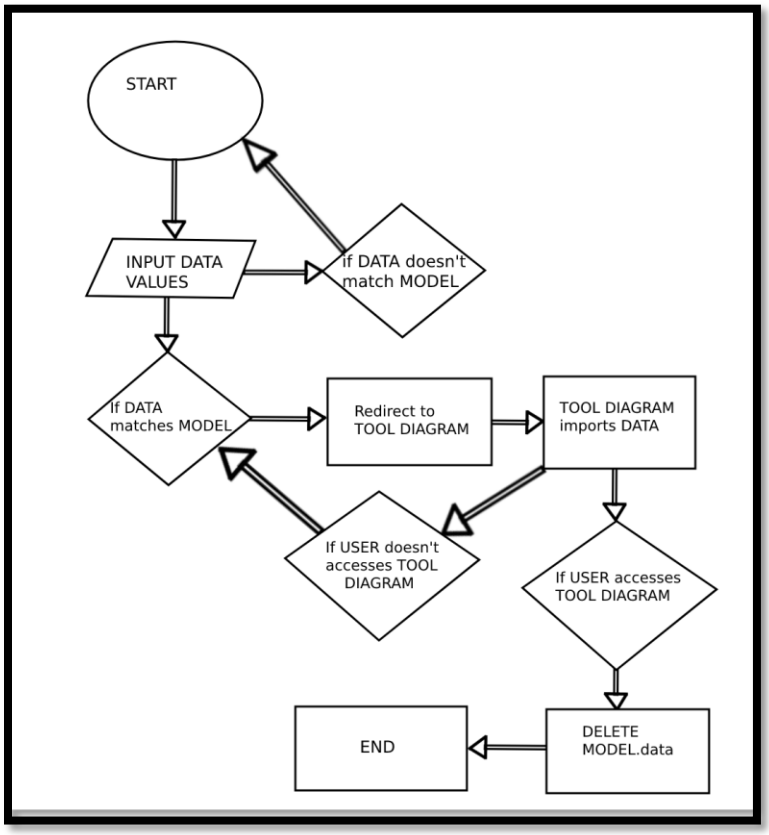
Decision Tree:

Figure 6: Decision Tree Diagram



Flowchart – Data management

Figure 7: Flowchart Data Management



Explanation of flow chart:

The process starts by requesting the user to input the data values for the business tool. My program then checks if the input data matches the MODEL (To check if it has correct characters, or any other extras). If no, then the process repeats, if yes, then it moves on. It will then redirect the user to TOOL DIAGRAM, and simultaneously will import the DATA into the TOOL DIAGRAM page. If the user was not able to access correctly the TOOL DIAGRAM page, then the program will again match the DATA to the model and re-try. If the USER accesses the TOOL DIAGRAM page, then the DATA can be automatically erased from the model since it isn't required anymore. This will clear up more space for future DATA. Finally, the process ends.

Testing Plan

Action needing testing	**Test Method**	**Expected Result**
If the website can run properly	Attempt to execute website on a server.	For the website to run with no error.
Admin login to database	Attempt to create an admin login	Be able to log-in as an admin into the database
Main page buttons	Get 3 users to attempt to use the buttons.	Redirect to correct website.
Force-Field Diagram Tool	Launch tool on another computer to make sure it functions everywhere.	Force-Field Diagram with correct values.
Break-even Analysis Tool	Launch tool on another computer to make sure it functions everywhere.	Break-even Analysis with correct values.
Graphing Tool	Launch tool on another computer to make sure it functions everywhere.	Graph with correct values.
Decision Tree Tool	Launch tool on another computer to make sure it functions everywhere.	Decision Tree with correct values.

Record of Tasks

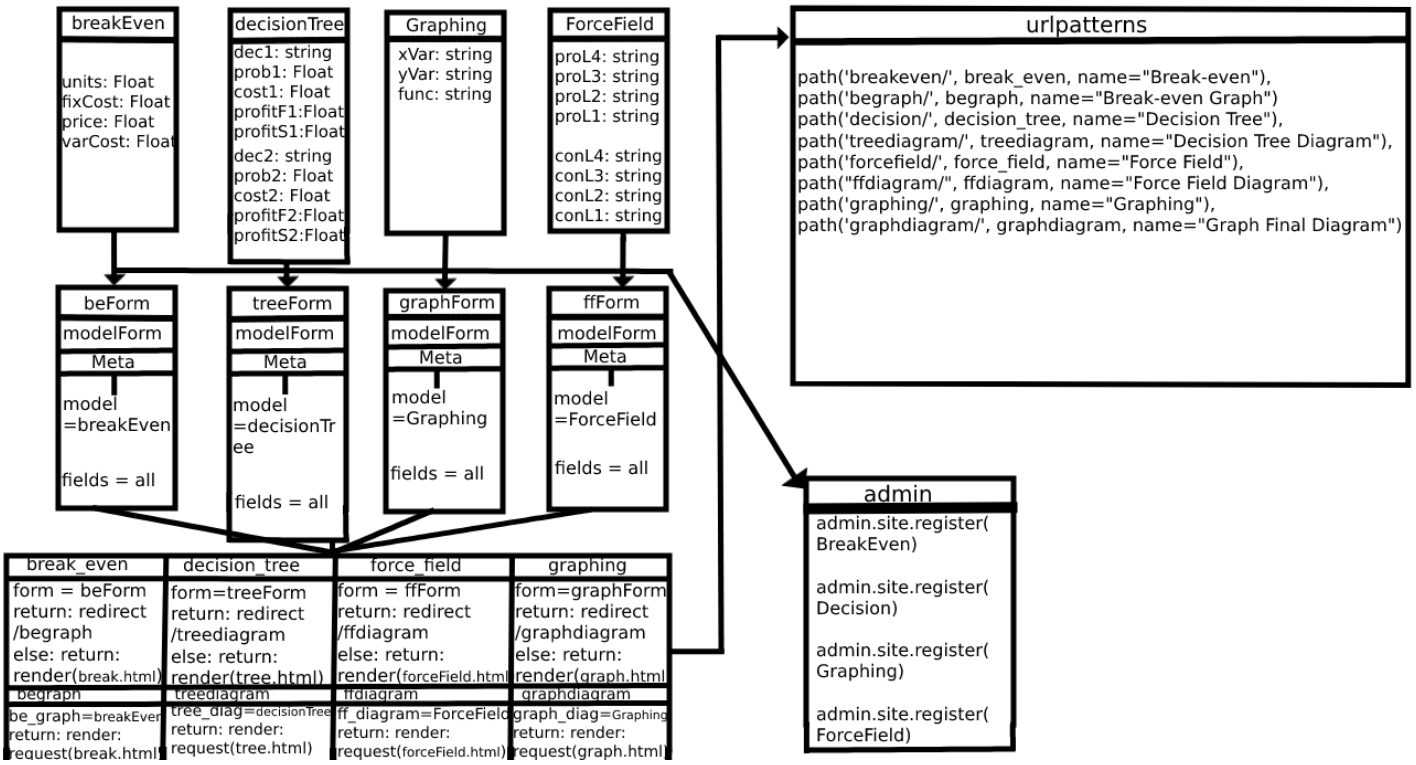
Order	Task Action	Estimated Time	Target Date
1	Consult Client	1 Hour	17/08/2022
2	Consult Advisor	30 Minutes	19/08/2022
3	Create Plan	1 Hour	19/08/2022
4	Create main page	2 Hours	10/09/2022
5	Create models & database	1 Hour	16/09/2022
6	Create first tool	2 Hours	17/09/2022
7	Create remaining tools	4 Hours	20/11/2022
8	Product Testing	2 Hours	01/12/2022
9	Final evaluation	1 Hour	05/12/2022

330 Words

Criterion C

UML Diagram

Figure 8: UML Diagram



The UML Diagram above represents and shows the process that the Models of the data go through from server-side to client side. It contains functions, variables, fields, and classes.

Techniques used

N.	Complexity Level	Techniques
A.	1	URLs
B.	2	HTML if statements
C.	3	Models
D.	4	HTML Redirects
E.	5	INPUTS and Forms

A. URLs

URL organization was an extremely important part of my product. Making sure all the URLs were connected to the correct .html files and that they executed properly. As well as making sure redirects went to the correct URL.

URLs are mainly organized through 1 python file which I called “urls.py”. This file serves as a list of all the URLs as well as their foundation. We can observe below this list.

```
urlpatterns = [
    path("", home_view, name="Home Page"),
    path('admin/', admin.site.urls),
    path('breakeven/', break_even, name="Break-even"),
    path('begraph/', begraph, name="Break-even Graph"),
    path('decision/', decision_tree, name="Decision Tree"),
    path('treediagram/', treediagram, name="Decision Tree Diagram"),
    path('forcefield/', force_field, name="Force Field"),
    path("ffdiagram/", ffdiagram, name="Force Field Diagram"),
    path('graphing/', graphing, name="Graphing"),
    path('graphdiagram/', graphdiagram, name="Graph Final Diagram")
]
```

I chose this way of organizing my URL’s since it is the most suitable when it comes to operating with Django. “urlpatterns” is a variable with a list fitting all the URL paths. The “path” tool is a Django keyword which sets up a route for my website. It starts with a string containing the URL name, for example “breakeven/”. Then it is followed by the views function which it will route towards. Then it is followed by a “name” keyword which contains a string, this string defines the name of the site which the URL contains.

Most of this was learnt through docs.djangoproject.com (All proper bibliography at the end)

B. HTML if statements

HTML if statements are almost as simple as Python if statements. They are a bit different since they require to be enclosed by braces (wavy brackets), they also need an “endif” to finalize the if statement. I utilized this mainly to produce my Force Field Diagram, a part of it can be seen below.

```
<strong><font size="+1">Pros:</font></strong>
<br>
{% if ForceField.proNumber4 > 0 %}
    Level 4: {{ ForceField.proLevel4 }}
<br>
{% endif %}
{% if ForceField.proNumber3 > 0 %}
    Level 3: {{ ForceField.proLevel3 }}
<br>
{% endif %}
{% if ForceField.proNumber2 > 0 %}
    Level 2: {{ ForceField.proLevel2 }}
<br>
{% endif %}
{% if ForceField.proNumber1 > 0 %}
    Level 1: {{ ForceField.proLevel1 }}
<br>
{% endif %}
```

Here it is utilized to know if the user has input certain variables or not, if the variable “proNumber#” is = 0 then the variable “proLevel#” will not be shown on the website, if it is higher than 0 then it will. This technique is useful since it allows me to link my knowledge to Python which I am more familiar with and function more efficiently in.

C. Models

The models I created start to get a bit more complex, but even more important for my code. Models are a definitive source of information surrounding data. It will contain the values I am storing. I created 4 Models, one for each business tool. Under we can observe an example of one, which is the Force Field Diagram model.

```
class ForceField(models.Model):
    busName = models.TextField()

    proLevel4 = models.TextField(blank=True)
    proNumber4 = models.IntegerField(blank=True, default=0)
    proLevel3 = models.TextField(blank=True)
    proNumber3 = models.IntegerField(blank=True, default=0)
    proLevel2 = models.TextField(blank=True)
    proNumber2 = models.IntegerField(blank=True, default=0)
    proLevel1 = models.TextField(blank=True)
    proNumber1 = models.IntegerField(blank=True, default=0)

    conLevel4 = models.TextField(blank=True)
    conNumber4 = models.IntegerField(blank=True, default=0)
    conLevel3 = models.TextField(blank=True)
    conNumber3 = models.IntegerField(blank=True, default=0)
    conLevel2 = models.TextField(blank=True)
    conNumber2 = models.IntegerField(blank=True, default=0)
    conLevel1 = models.TextField(blank=True)
    conNumber1 = models.IntegerField(blank=True, default=0)

    def __str__(self):
        return self.busName
```

This is by far my biggest model, since I had many different variables. This model is stored in my admin database. It is initialized using a class with an instance of “models.Model” which is a Django property. Then I can create various variables inside this Model where data will be stored. This variable is set to be the keyword “models.” followed by the field type. Then inside the parentheses in some cases I had to put “blank=True” which simply means that it is not obligatory for that variable to be fulfilled. For any integer fields it’s also required to put “default=0” inside the parentheses since integer fields give an error if null. Using models to organize my data is perfect since I can separate different tools into their own area without it getting too confusing, it also allows me to directly fetch certain variables.

This information came to me through the source docs.djangoproject.com

D. HTML Redirects

HTML Redirects are used in my views and HTML files to simply send users to another website after a certain function has occurred. It is a tool that comes with Django. I employed HTML redirects mainly after the user submits the data for a business tool. It is short but can be complicated to understand.

```
def force_field(request):
    submitted = False
    if request.method == "POST":
        form = ForceFieldForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("/ffdiagram")
    else:
        form = ForceFieldForm
        if "submitted" in request.GET:
            submitted = True
    return render(request, "forceField.html", {"form": form, "submitted": submitted})
```

This is inside of the view function for all my tools, the HTML Redirect will redirect the user through the keyword “HttpResponseRedirect” through an if statement. The return keyword makes it the final executable line of code inside the if statement. Inside the parentheses the URL which the client will be redirected to will be placed. HTML Redirects are important in my project since users will need to be automatically redirected to the site containing the executed business tool, “HttpResponseRedirect” is the easiest technique to do so since it redirects the user with a single line of code.

A source which helped me with this was realpython.com

E. INPUTS and Forms

INPUTS on the site and Forms were by far the most complex code I had to create in this project. My idea was to have a site where users could input data which is then stored in the database and later fetched by the website to create the business tool diagram. Firstly, I had to create a form on a python file utilizing the Django framework.

```
class BreakEvenForm(ModelForm):
    class Meta:
        model = BreakEven
        fields = "__all__"


class DecisionForm(ModelForm):
    class Meta:
        model = Decision
        fields = "__all__"


class ForceFieldForm(ModelForm):
    class Meta:
        model = ForceField
        fields = "__all__"


class GraphingForm(ModelForm):
    class Meta:
        model = Graphing
        fields = "__all__"
```

The forms is the simplest part. It is simply a class for each business tool with the instance “ModelForm” by Django. Then I created a subclass called Meta, which is used to change the behavior of the model fields. Inside I call the model corresponding to the business tool and call all available fields (Integer, String, etc...).

```python
def force_field(request):
    submitted = False
    if request.method == "POST":
        form = ForceFieldForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("/ffdiagram")
    else:
        form = ForceFieldForm
        if "submitted" in request.GET:
            submitted = True
    return render(request, "forceField.html", {"form": form, "submitted": submitted})
```

Then in the views file, I call the Form as a variable “form”. Then the client is given the form in the site to input required data. Once data has been input and submitted the if statement will validate and redirect the user to the diagram.

```html
<center>
    <form action="" method=POST>
        {% csrf_token %}

        {{ form.as_p }}

        <input type="submit" value="Submit">
    </form>
</center>
```

Inside the HTML file it is much simpler, and you simply call the form through double braces by calling the “form” variable. In the views function the variable has already been sent to the html file.

This technique is complex but the easiest way to have users input data in Django. Utilizing this helped me immensely since my website would not reach success criteria without personalized user input data.

As sources for this I utilized geeksforgeeks.org and docs.djangoproject.com

Word Count: 1001

Bibliography

https://www.geeksforgeeks.org/meta-class-in-models-django/#:~:text=Model%20Meta%20is%20basically%20the,Meta%20class%20to%20your%20model.

https://docs.djangoproject.com/en/4.1/topics/forms/

https://realpython.com/django-redirects/

https://docs.djangoproject.com/en/4.1/topics/db/models/

https://docs.djangoproject.com/en/4.1/topics/http/urls/#naming-url-patterns