GE## Documentation

Jacob Hartt, Takaiya Jones, Jacob Biles, and Zachary Rytting

Professor Deb Harding
CS3300.002
MM.DD.YYYY

0.0 SECTION TEMPLATE

The homepage is the first display seen by users when coming to a site. Homepages are an entryway to expectations, it is important to build and comprehend the details of your site.

Issues Encountered

- Issue 1: The homepage utilizes different aspects of software concepts and principles to understand models, views, HTML styling, and other components crucial for sites. Before you even start designing and setting displays on your screen, you need to understand the relationship between different areas in your code. The models created earlier contain certain characteristics that describe what it is. Your models are needed when referring to the “views.py”.Functions inside of “views.py” create variables that search for specific requirements inside of the models listed earlier. The function then returns the item searched and

directs this response to the HTML template. The HTML template is like a skeleton of what the website will display.

- Solution 1: Take time to understand the code you produce before moving on to the next topic. When creating a model, we establish characteristics with needs. Each model contains needs.

```
class Project (models.Model):
    title=models.CharField(max_length = 200)
    description=models.TextField()
    #description=models.TextField(blank=True)
    #portfolio = models.OneToOneField(Portfolio,nu
    portfolio = models.ForeignKey(Portfolio, on_de
```

- Afterward, we have a view section that searches through the database for required information and saves that information. The information is sent to an HTML link, which will display the required information from active students.

```
def index(request):
    student_active_portfolios = Student.objects.select_related('portfolio').a
    print("active portfolio query set",student_active_portfolios)
    return render(request,'portfolio_app/index.html',{'student_active_portfol
```

- The “ursl.py” contains all the paths to where the user is led. The “urls.py” and “views” work together to establish routing. Without the URL paths, would not be able to navigate through the site. Be sure when a new page is created to link it in “urls.py”.

```
from django.urls import path
from . import views

urlpatterns = [
    path('',views.index, name='index'),
    path('portfolio/<int:pk>',views.PortfolioDetailView.as_view(),name='portfolio-detail'),
```

- The HTML document customizes the appearance and displays the required data, active students, on the homepage. A “for loop” iterates across the

active student and matches correlating data.

```
{% for i in student_active_portfolios %}
    <li class = "list-group-item">Name: {{ i.name }}

        <p><strong> Portfolio:{{ i.portfolio.title }}
```

- The last step is the creation of the button. The for loop gets the portfolio specified portfolio and links it to the next page when the button is clicked. Earlier we created a function inside the models named “get_absoulute_url”: the “get_absoulute_url” function guides the user to more information on the specified portfolio.

```
        <a href= "{{i.portfolio.get_absolute_url}}"> <button> View </button> </a>
```

- The “ursl.py” contains all the paths to where the user is led. The “urls.py” and “views” work together to establish routing. Without the URL paths, would not be able to navigate through the site. Be sure when a new page is created to link it in “urls.py”.

```
from django.urls import path
from . import views

urlpatterns = [
    path('',views.index, name='index'),
    path('portfolio/<int:pk>',views.PortfolioDetailView.as_view(),name='portfolio-detail'),
```

Useful Resources

- [Resource 1]
  - This site shows the bare bones of creating a homepage. It is a helpful guide in creating a server.

0.1 SECTION TEMPLATE

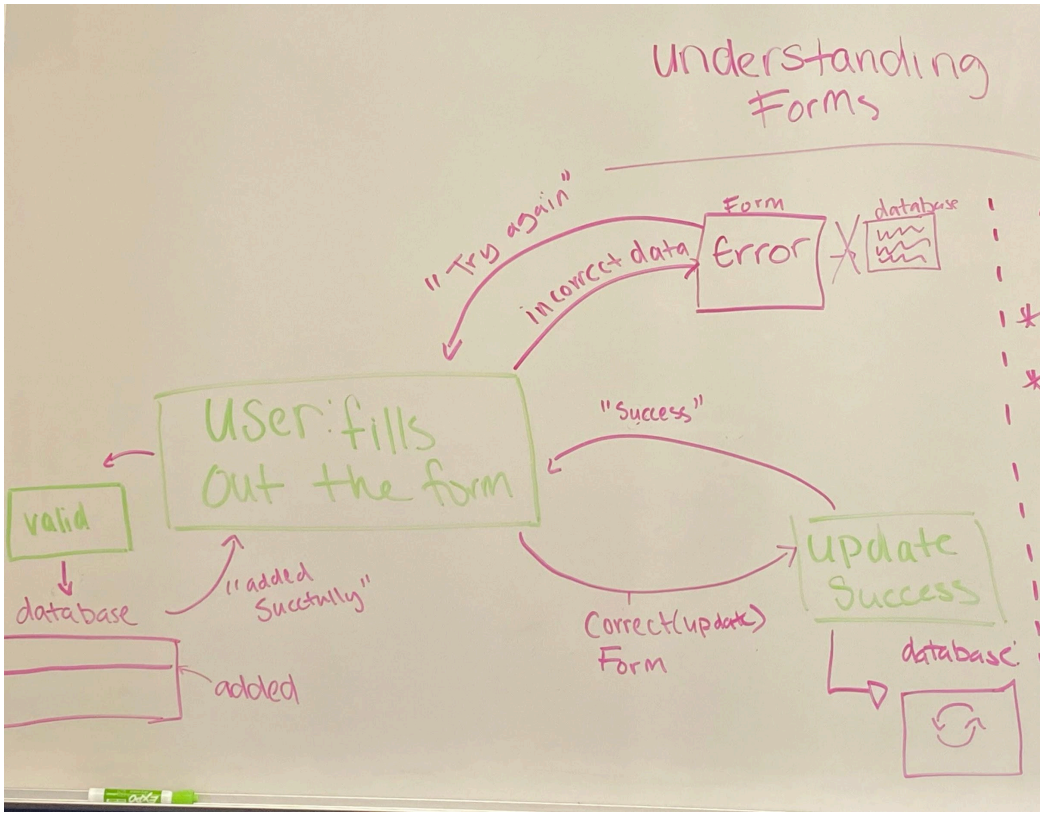
Forms are one methodology for taking user input. Towards the end of the project, the focus is on updating and creating new forms that will aid in gaining incoming user

information. Creating and updating forms is an important aspect of taking user data and applying the changes to the database.

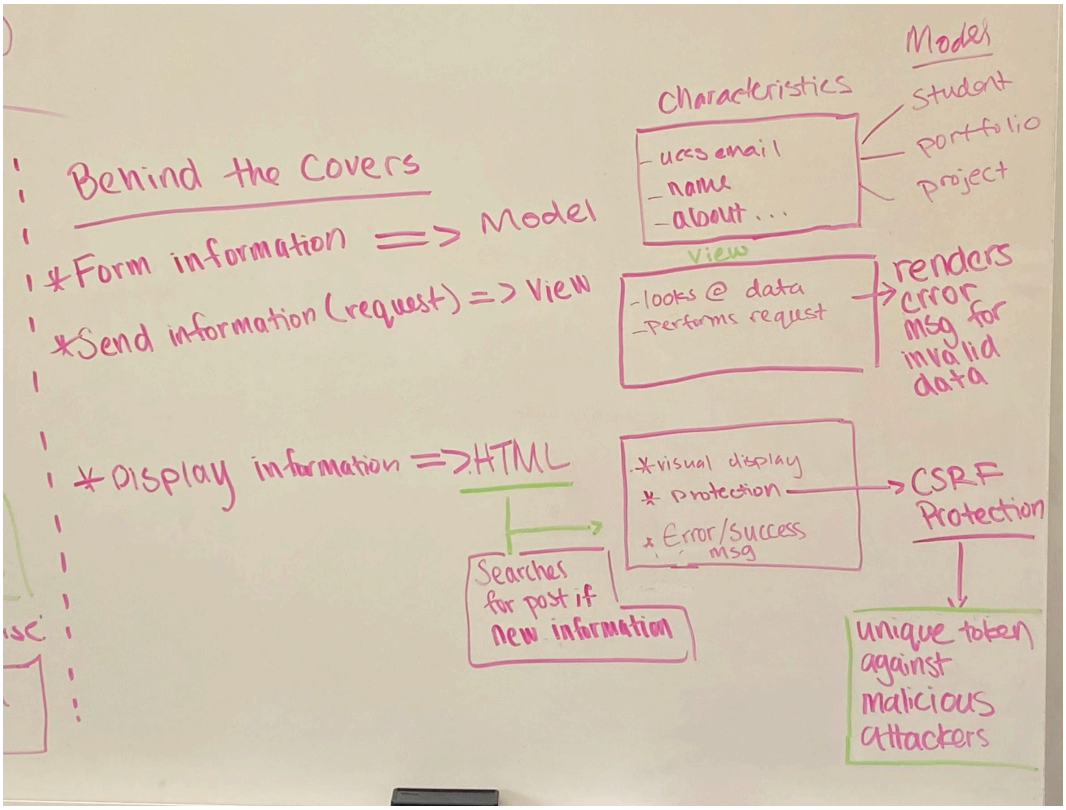
Issues Encountered

- Issue 1: Although filling out forms is a simple task, understanding how they work is challenging. When forms are submitted by the end-user, a “POST ” is made. The form checks for correct date entry that then edits the database, but disregards insufficient. It is complicated to visualize how forms are produced.

- Solution 1:
  - Pictures/diagrams are an amazing way to visualize confusing concepts and break them down into more manageable tasks. To understand the process of creating a form, draw a picture.
    - A form requires information. The model provides the information for the form requirements. For example, the user must enter a title and description to move forward. A valid response involves filling out the necessary data, while an invalid response produces an error message. After the user fills out a valid form, a request is made and sent to “views”. The “views” searches and then executes a request. After the search, a response is returned. The visual display of the response is handled by the HTML, which includes a changing token to protect users.
    - On the other hand, if a user enters invalid data, they are given an error message from the “view.py”. The invalid data does not update or affect the database.


Understanding Forms
"Try again"
incorrect data
Form
Error
database
User fills Out the form
"Success"
valid
database
"added Succesfully"
added
Correct(update) Form
Update Success
database

➢


Behind the Covers
*Form information ==> Model
*Send information (request) => View
*Display information =>HTML
Searches for post if new information
Model
Characteristics
- user email
- name
- about ...
student
portfolio
project
View
-looks @ data
-Performs request
renders error msg for invalid data
*visual display
* protection
* Error/success msg
CSRF Protection
unique token against malicious attackers

Useful Resources

- Resource 1
  - This resource provides images and code snippets that explain the purpose of each aspect of form creation.