



MATES Computer Science

Senior Capstone Project Final Report

Project Title	Terrapin Tracker
Team Members	Angelo Amato, Aidan Henbest, Lisa Hunt
Link to GitHub	https://github.com/henbestaj/senior-capstone

I. Executive Summary

We have created a website where members of Project Terrapin can record the measurements of the hatchlings in their care. Project Terrapin is an organization that cares for and measures the growth of northern diamondback terrapin hatchlings at MATES and other affiliated schools. Using Python, the data collected from the turtle measurements will be displayed on different graphs and tables throughout the website. This website will be public and Project Terrapin members will be able to log in to record these measurements. The graphs and tables on the site will update automatically when new data is collected. Our project aims to raise awareness of the dangers that face the Northern Diamondback Terrapin species. They are a keystone species in the Barnegat Bay, meaning they are essential to the survival of the ecosystem. We hope our website will inspire others to get involved in the protection of terrapins, and let them track the growth of hatchlings they may have found, giving them a more personal connection to the turtle(s) they helped to save.

II. Detailed Summary

Overview

This project was chosen because Project Terrapin is a very prevalent club within our school, with a majority of students being involved with the club in some way, shape, or form. Angelo Amato in particular was the Conservation Coordinator for the 2022-2023 school year and has been involved with the organization since his freshman year. As previously mentioned, the northern diamondback terrapin is essential to the survival of the Barnegat Bay ecosystem; It is important to not only protect this species but to also spread awareness of its importance to the environment. This project is important because it will serve as a way for Project Terrapins to record their measurements on a publicly accessible website as opposed to a Google Sheet that can be edited by anyone and potentially lost at the end of the year when the district deletes the students' email addresses.

Technology and Versions

We used **Visual Studio Code** (Version 1.78.2) in order to code this website. The languages we used to program the site were **HyperText Markup Language** (Living Standard), **Cascading Style Sheets** (Snapshot 2023), and **Python** (Version 3.9.13). In addition we used many Python libraries:

asgiref (Version 3.6.0), cachetools (Version 5.3.0), certifi (Version 2022.12.7), charset-normalizer (Version 3.1.0), contourpy (Version 1.0.7), cssselect (Version 1.2.0), cssutils (Version 2.6.0), cycler (Version 0.11.0), Django (Version 4.1.7), fonttools (Version 4.39.3), idna (Version 3.4), importlib-resources (Version 5.12.0), kiwisolver (Version 1.4.4), lxml (Version 4.9.2), matplotlib (Version 3.7.1), mysqlclient (Version 2.1.1), numpy (Version 1.24.2), packaging (Version 23.0), pandas (Version 2.0.1), Pillow (Version 9.5.0), premailer (Version 3.10.0), pyarsing (Version 3.0.9), python-dateutil (Version 2.8.2), pytz (Version 2023.3), requests (Version 2.28.2), seaborn (Version 0.12.2), six (Version 1.16.0), sqlparse (Version 0.4.3), tzdata (Version 2023.3), urllib3 (Version 1.26.15), yagmail (Version 0.15.293), zipp (Version 3.15.0). Lastly, we used MySQL Workbench (Version 8.0.32) for our database.

Visual Studio Code was used to edit all of the files within the projects, including .py files, .html files, .txt files, .css files, .json files, .gitignore files, .md files, and .config files. The base of the website was built on the Python programming language, primarily utilizing the Django library (although many other Python libraries were also used). However, below even the base of the website the MySQL server was present, connected to the Python files. From the Python files, links on the website were associated with HyperText Markup Language files. From the HyperText Markup Language files, Cascading Style Sheets files were associated with pages on the website. Overall, the website structure went something like this: the MySQL server connects to Python files, Python files connect to HyperText Markup Language files, and HyperText Markup Language files connect to Cascading Style Sheets files.

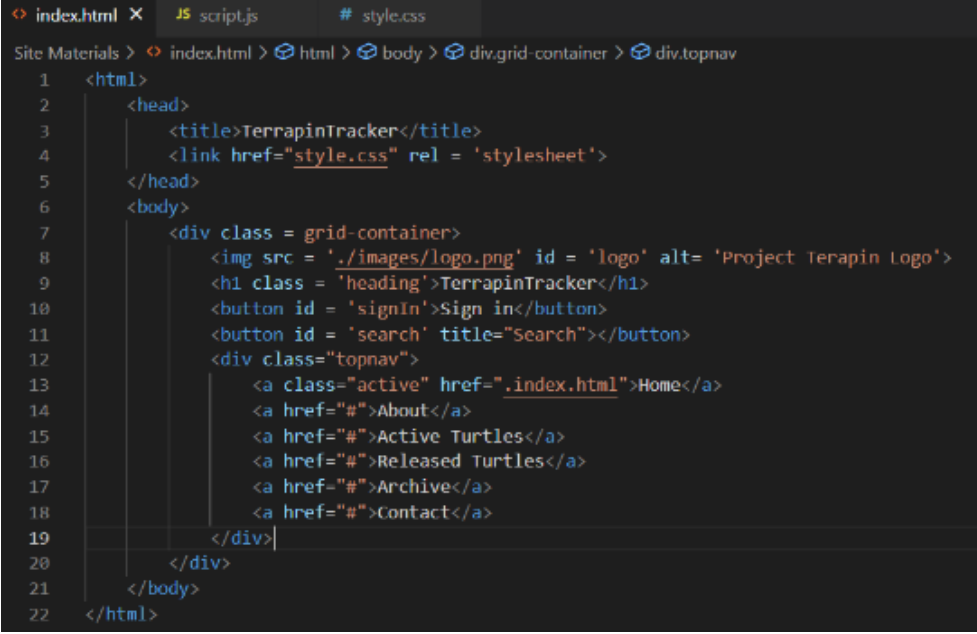
Key Milestones

The key milestones within the first two weeks, February 15 to February 26, consisted of learning the languages necessary to complete the project. Learn Django, HTML, CSS, and Javascript Codecademy courses were completed. We then gathered the measurement data from Project Terrapin members and organized it so it could be put into a SQL server.

In the month that followed, from February 27th to March 27th, we worked on creating all of the basic elements for the website. This included the base page with navigation links to the home page, about page, current page, released page, and contact page. Navigation buttons were created and all of the pages were linked together using Django. Initial CSS stylesheets and HTML files were created

during this time to bring color, style, and text to the pages. During this time as well, we worked on setting up the SQL server so it could be used to display the first measurement table on the website.

Base HTML - Early March



```
1 <html>
2   <head>
3     <title>TerrapinTracker</title>
4     <link href="style.css" rel = 'stylesheet'>
5   </head>
6   <body>
7     <div class = grid-container>
8       <img src = './images/logo.png' id = 'logo' alt= 'Project Terapin Logo'>
9       <h1 class = 'heading'>TerrapinTracker</h1>
10      <button id = 'signIn'>Sign in</button>
11      <button id = 'search' title="Search"></button>
12      <div class="topnav">
13        <a class="active" href=".index.html">Home</a>
14        <a href="#">About</a>
15        <a href="#">Active Turtles</a>
16        <a href="#">Released Turtles</a>
17        <a href="#">Archive</a>
18        <a href="#">Contact</a>
19      </div>
20    </div>
21  </body>
22 </html>
```

In the next period from March 28th to April 6th and April 15th to April 25th, we continued creating various pages for the website. Both the contact and about pages were completed during this time. The current page expanded to include R group buttons with tables and graphs of the measurement data. We created user login abilities with an account system for Project Terrapin members to use to edit turtle information. The home page graphs were also completed during this time.

In the last month of the project, we completed all of the pages and created edit pages, delete pages, history pages, and archive pages. As new HTML pages were created, new CSS files were also created to style the elements. Within the last two weeks, we worked on styling all of the Django forms that users enter information. We also created errors for all forms, so if users input wrong information, Django will trigger an error.

Scope Changes

The scope of our project was changed many times, as we established certain features that were not necessary for the final product. One of the earliest changes was our decision not to use D3 graphs. D3 could have enabled us to create interactive graphs for the website, but we decided that learning to implement that style of graph would not be worth the time. We attempted to create D3 graphs a few times but they would not display on the website. We already knew how to make graphs using Seaborn and they were much easier to display on the website, so we decided against using D3, even if it meant losing the interactive features of D3.

Another change we had to make was when we received new information from Project Terrapin members on how they collect hatching measurements. We learned that they do not keep track of individual hatching numbers but only the R group, or physical tank that the turtles live in. Our original plan was to create turtle profiles for each hatching that showed their measurement history, but since the data mixed around ID numbers, we switched over to only displaying R group numbers on the website. This meant that all of the graphs showed data from the turtles within the R group instead of the individual turtles.

As we were building our website, we realized somewhere down the line that we did not need to implement Javascript anywhere. We had completed the Codecademy course, but decided against using Javascript because we were not well-versed in the language due to lack of use in the website. This was another part of deciding against D3, which is coded in Javascript.

Towards the end of the project, we excluded hosting the website from our final goals. For the sake of the project, we wanted to make sure all of the website features were perfect before publishing, and we ran out of time to host the website. We do plan on hosting the website later this month, however, so Project Terrapin can access it for next year.

We had a few other minor scope changes, such as not using the search button as an input field. We originally wanted it to function as a box where users could enter a turtle ID and it would link to that page. However, we found it easier to create an HTML search page instead and use a Django form for the search feature. Lastly, when loading the website's home page, there would occasionally be an error that would resolve when reloading the page. Since we didn't know what the error was or how to trigger it we decided not to fix it especially since it seemed to fix itself.

Next Steps

When it comes to the next steps for continuing this project in the future, the primary goal is to upload the website to an Amazon Web Services (AWS) Server and have a presence on the web. This also includes gaining access to a Uniform Resource Locator (URL) for the project. We will be continuing to pursue this goal over the summer, so Project Terrapin can use this website next year. Besides this AWS server, there are not many other next steps that we plan on pursuing as we achieved most of our other goals. The only other next step we have considered is linking our website to the main Project Terrapin website, but as we looked further into this, it became apparent that it was practically impossible as coming into contact with the person who updates this website is difficult.

Individual Contributions

The work was divided in a fairly simple manner: Aidan Henbest was the primary backend developer, while Angelo Amato and Lisa Hunt mainly focused on the front end. Of course, overlap often occurred between these two groups (i.e. Lisa created the graphs on the backend before working on displaying them on the frontend), but for the most part, each member of the group stayed in their own lane.

Lisa:

In the first two-week period, I worked on learning CSS and HTML on Codecademy. In CSS, I learned how to display elements and position them on the page. I also learned how to create website buttons that function as links. I learned how to style websites using color, font, white space, photos, and more. On the website, I helped with the grid formatting for the navigation bar and designed the search and sign-in buttons. I fixed the position of the buttons within the grid so they would align with the title and logo. I added clickability features to all of the buttons, meaning that when the mouse hovers over them, the color and style will change to indicate to the user that it is a button. I also learned Javascript basics through Codecademy during this time.

In the next two weeks, I worked on learning Django and how to implement it into CSS and HTML files. I then worked on the search box, making it expand and function as an input box. However, we ended up scrapping this part of the search button later. I also started working on the Current Turtles page. I formatted the initial temporary table using a flex display, so it would fit on the page. I started

Current Table - Late March

R1	R6	R7	R8	R9
R10	R11	R12	R13	R14

During the next period, in the last two weeks of April, I worked on creating the graphs for the current turtles page. During this time, we figured out that we would need to use R groups instead of individual turtle IDs as explained above. This meant that all of the graphs needed to sort the measurements by R group to display. We also decided against using D3 at this point and started to create graphs using matplotlib. Matplotlib had an issue with displaying dates on the graphs, so I switched to using Seaborn for all of the plots. I created all of the current turtle graphs during this time. The data for the graphs is the measurement data sorted by R number and put into lists. I first graphed carapace length and width, using a scatter plot and kde plot. I then did the same for plastron length and carapace height. I then made box plots for carapace length and height over time. I made a histogram showing the mass distribution and then a bar plot of the average mass by turtle number. I saved all of the graphs to their own file path, which I could save to the static folder and add to the HTML page. I then went back and styled all the graphs, changing size, color, and labels. I also styled the measurement table during this time. At the end of April, I styled the measurement history table and a few other minor elements such as the welcome message, contact sent page, edit buttons, and back buttons.

```
# creates carapace length by carapace width scatterplot
fig, ax = plt.subplots()
sns_plot = sns.scatterplot(ax=ax, x=carapace_length, y=carapace_width, hue=date).set_title('Carapace Length vs Carapace Width')
ax.set_xlabel( "Carapace Length" , size = 12 )
ax.set_ylabel( "Carapace Width" , size = 12 )
file_path = './turtles/static/turtles/plot_r' + str(r_num) + 'year' + str(year_archived) + 'lengthvswidthscatter.png'
fig = sns_plot.get_figure()
fig.savefig(file_path)
plt.clf()
plt.close()
```


In the first two weeks of May, I went back to the current turtles table and figured out the format. I ended up using Django to loop through the data and a grid template using auto flow, meaning the grid will update once a row of 5 turtles was filled. Now the grid could account for new R numbers and put them in order on the page. I then made all of the home page graphs. For this I had to create a new set of lists that sorted the data by R group. The graphs I made were similar to the R number graphs, with a few variable swaps to give a more general sense of the data. I then sorted the legend for the graphs that it applied to and made sure it would work for new R groups as well. Next, I created an alert box for the login message. This means that it will pop up on the screen once the user is logged in and then the user can click to remove the message. I also added the settings button and formatted it similar to the search button. I also did a lot of styling to buttons throughout the website and minor pages like the contact sent message. I formatted the released turtles page so when data exists next year, it will display with a similar format to the current turtles page sorted by year. Next, I spent a lot of time fixing the errors on the sign in page, since I had to account for multiple errors triggering at the same time. Lastly, we have had a horizontal scroll bar on all pages for a long time, so I finally got rid of it.



In the last two weeks of the project, I worked on styling errors for all of the Django forms across the website. The included messages for incorrect user inputs like negative numbers, mismatched passwords, nonexistent turtles, and more. I ran into a few issues when trying to get the error messages to display correctly, but I was able to fix everything. I continued to style any remaining buttons and page elements. I added in the home page description and code comments on all of the HTML and CSS files to complete my part of the project.

Angelo:

I spent the first two weeks of the project helping to determine the scope of the project, as well as learning the necessary languages. I worked primarily with Codecademy courses. I used the lessons from the Full-Stack Engineer Career Path on the site. I learned HTML, CSS, and Javascript in preparation for use in this course as well as how to locally run a website. The HTML and CSS were useful in designing our website. In CSS I learned how to use grids to format HTML elements.

Furthermore, I have started working on a Django course in order to learn how to use templates, link pages together, and work with the back end as well. Most of the time spent on the project itself was dedicated to creating the home page and the header that will be used on all of the pages in the site for navigation. The color palette we used was adapted from the Project Terrapin website. I inserted the Project Terrapin logo in the top left corner, however, we have created a new logo to replace this. The navbar changes to let the user know that the elements are clickable and can be used to change the pages of the website.

```
<!-- creates navigation bar for all website pages -->
<div class="topnav">
  {% with alert=0 %}
  <a class="{% home_act %}" href="{% url 'home' alert %}">Home</a>
  {% endwith %}
  <a class="{% about_act %}" href="{% url 'about' %}">About</a>
  <a class="{% current_act %}" href="{% url 'current' %}">Current Turtles</a>
  <a class="{% released_act %}" href="{% url 'released' %}">Released Turtles</a>
  <a class="{% contact_act %}" href="{% url 'contact' %}">Contact</a>
</div>
</div>
```

In the next progress period, I worked to create various pages for the website. I created the HTML files and some of the corresponding CSS files. With Aidan's assistance, I used Django to link the pages together using the nav bar. A decent portion of my time this period was spent attempting to load the server in my browser using Django. This set the tone for this period as I spent a decent amount of time familiarizing myself with the programs. Many of my errors were caused by my lack of familiarity with the tools we used. For example, I did not realize that the browser would cache the site and not update with new changes. In regards to the pages I created this period, I mainly worked on the about us page. I wrote a brief description of our project and inserted "cards" of all of us. Included in these cards are information about us, our emails, and personal GitHubs.

In the next two weeks, I primarily worked on styling the pages. I started off by making the sign-in page actually work (changed from a blank page). I used Django's functionality for creating a login system. Aidan helped me with getting the actual page to work with the buttons (Django has a built-in URL path that I did not know how to integrate). I then had to figure out how to style the input boxes even though they are not correlated with the actual HTML in the file. All Django forms are given a specific ID that can be found by inspecting the web page. I used these to style the actual log-in page.

I added a white background box to both the log-in and contact forms to make them stand out. I then moved on to working with the forms that actually log the turtle's measurements and created new turtles. I completed work on the "create turtle" page and designed the "create measurement" page.

```
class LoginForm(forms.Form):
    def clean(self):
        data = self.cleaned_data
        username = data['username']

        if User.objects.filter(username = username).exists() == False:
            raise forms.ValidationError("Please enter a correct username.")

    username = forms.CharField(label='Username')
    password = forms.CharField(label='Password', widget = forms.PasswordInput)
```

I started off the next period by finishing the measurement page. I had some issues with this as the form elements are contained within a <p> tag (Django feature). Working with the grid was a little difficult because of the unusual HTML structure. I used CSS grid to style this page, but unlike the "new turtle" page, I had to use two different columns so all of the inputs would fit nicely. Moving on I began to style the "Sign Up" page which was a bit easier because I had gotten used to styling the Django forms, but again making two different columns for the inputs had a few kinks that needed to be worked out. I then moved on to styling the settings page, which was the most difficult page so far. This is because the page contained 3 different forms to reset passwords, reset usernames, and delete accounts. I put all of the forms in a container element and then into separate divs from there. I made the divs that contained each individual form darker than the ones that contained all three. Ultimately I used Grid to move the forms into place, and flex to style the inputs and headings within the forms.

```
<div id = "center-box">
  <h1 id= heading>Sign Up</h1>
  {% if user.is_authenticated %}
  <p class="log">Please log out if you would like to create a new account.</p>
  {% else %}
  <h2 id="personal-info">Personal Info</h2>
  <h2 id="set-password">Set Password</h2>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <!-- puts errors into list for styling -->
    {% if form.errors %}
    <ol class = error_list>
      {% for key,value in form.errors.items %}
      <li class="error">{{ value|escape }}</li>
      {% endfor %}
    </ol>
    {% endif %}
    <!-- form will not submit and trigger an error if one of the conditions is not met -->
    <ul class="form"><li>Your password can't be too similar to your other personal information.</li></ul>
```

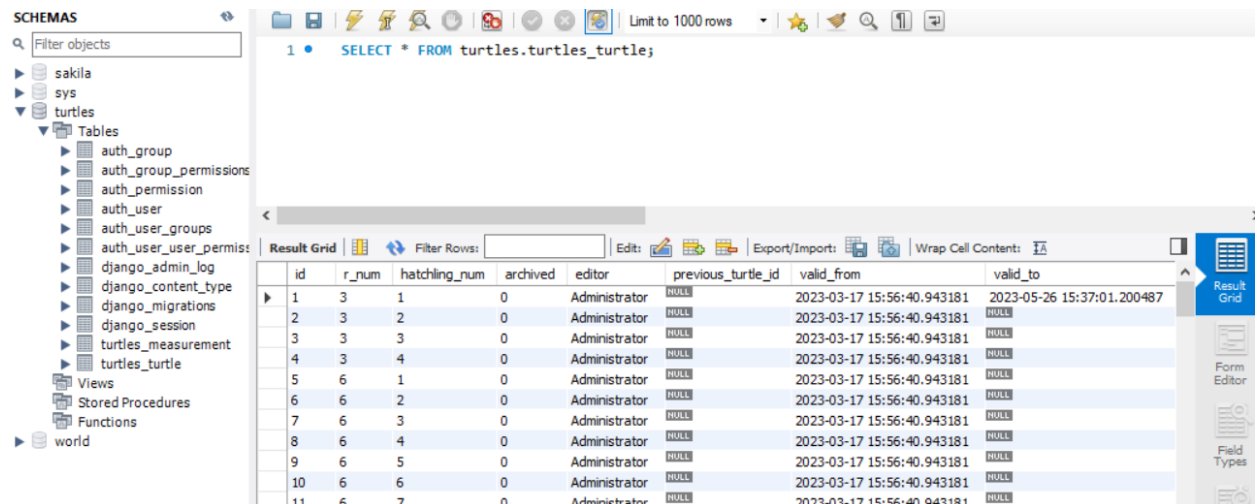
In the most recent progress period, I worked on styling the various new features that were put into the website along with finishing touches. I formatted the form to edit measurements and turtles. I also styled the mass turtle create and mass archive pages, which will be used to expedite the process of creating and releasing turtles. This was difficult because of how Django created the checkboxes on the archive page. I also formatted the pages used to log the measurement edit history and the turtle measurement history. Finally, I styled the confirmation pages for when the turtles are deleted, making sure the user doesn't accidentally remove a turtle still in Project Terrapin's care.

Aidan:

As the primary backend developer, the first step was to learn how to use Django, which is the Python-based framework we used to create the website. To achieve this I completed the Build Python Web Apps with Django Skill Path on Codecademy. This course was very useful and I couldn't have completed the project in such an efficient manner without it. After the course was done, I had to start the Django project and app using Powershell. Furthermore, I also created a Python virtual environment to use for this project. Next, I had to edit the automatically created settings.py file in order to prevent the private Django secret key from uploading to GitHub. To achieve this, I inserted the secret key into a new file, imported the variables from the new file into the original file, and

added the new file to a .gitignore file. I also created many other .gitignore files in order to exclude the virtual environment among other things.

The next step that I took while creating the website was working on the MySQL server. In order to achieve this I had to create the Django models (which translate to MySQL tables), including all of the columns and methods associated with them. Two models were created, the Turtle model and the Measurement model. Each row of the Measurement model links to a row of the Turtle model. After creating these models, I had to run a few scripts in Powershell in order to get Django to create the tables in the MySQL server. Lastly, in MySQL Workbench I needed to import the hatchling data given to us by Project Terrapin from this year (after formatting the data). I ran into many problems while doing this, including the fact that some of the columns I had created weren't allowed to be blank or didn't have a default value. However, I resolved all of these issues after a day or two.



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the database structure, including tables like 'auth_group', 'auth_group_permissions', 'auth_permission', 'auth_user', 'auth_user_groups', 'auth_user_user_permissions', 'django_admin_log', 'django_content_type', 'django_migrations', 'django_session', 'django_measurement', 'turtles_turtle', 'Views', 'Stored Procedures', and 'Functions'. The main pane shows a query: `SELECT * FROM turtles.turtles_turtle;` and the resulting data grid.

id	r_num	hatchling_num	archived	editor	previous_turtle_id	valid_from	valid_to
1	3	1	0	Administrator	NULL	2023-03-17 15:56:40.943181	2023-05-26 15:37:01.200487
2	3	2	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
3	3	3	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
4	3	4	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
5	6	1	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
6	6	2	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
7	6	3	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
8	6	4	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
9	6	5	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
10	6	6	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL
11	6	7	0	Administrator	NULL	2023-03-17 15:56:40.943181	NULL

After this, I worked on a form and associated views that allow the users of the website to create new turtles and measurements. This task was fairly easy, as Django has many built in functions for allowing this ability. However, that isn't to say there weren't any issues. For example, at first when the user was selecting a turtle from the dropdown when creating a measurement, all turtles would appear, even ones that had been archived or deleted. I was able to fix this by overriding the `__init__` method of the form. Another issue I encountered was figuring out a way to prevent the user from including negative numbers in their measurements. This was simple for the turtles, as the fields of the create turtle form were positive integer fields; however, there is no positive float field that I could use for the create measurement form. Therefore, I had to figure out how to override the

default clean method on the form and throw errors to the user. This was very helpful later as I created more and more forms.

Next, I worked on allowing the user to contact superusers (or administrators of the website) through email. To do this I created a form that allowed the user to enter their email address, a subject line, and an email body that will associate with a view that sends an email to administrators. This was difficult, to achieve it I had to learn how to use the Python library Yet Another Gmail (or yagmail for short) and create an email to use for the website (terrapintrackercontact@gmail.com). I then had to associate the Gmail account with the Python file using OAuth 2.0 credentials among other Powershell prompts.

```
def contact(request):
    if request.method == 'POST':
        form = NewContactForm(request.POST)

        if form.is_valid():
            emails = []
            for i in User.objects.filter(is_superuser = True):
                emails.append(i.email)
            yag = yagmail.SMTP('terrapintrackercontact@gmail.com', oauth2_file = './oauth2_creds.json')
            yag.send(to = emails, subject = str(form.cleaned_data["subject"]), contents = 'From:\n' + str(form.cleaned_data["email"]))
```

Another main task I worked on was the log in, log out, and sign up pages. Each of these pages appeared to be easy to build because of Django's built-in functions, but in reality, it was very difficult. Logging in and logging out were easy; however, the sign up page was a struggle. Creating a basic Django sign up page is simple, but we wanted to add one big addition: email verification. First, we wanted to force the user signing up to use a @ocvts.org or @mail.ocvts.org email address. Secondly, we wanted to verify this address by sending a code to their email. Both of these tasks proved hard, but I ultimately was able to achieve both of them. The main issue was storing the code sent to the user in a secure manner, but by using a session variable I was able to do it.

Next, I took up a fairly simple task: allowing users to download the data on the website in a .csv file. This was fairly simple once I learned how to create a .csv file using Python, and I created one button that downloads a .csv file of the current measurements and another that downloads a .csv file of the deleted measurements.

```
filename = './turtles/static/turtles/current_measurements.csv'
download_name = 'current_measurements.csv'
wrapper = FileWrapper(open(filename))
response = HttpResponse(wrapper, content_type='text/csv')
response['Content-Disposition'] = "attachment; filename=%s"%download_name
return response
```

The next set of tasks took a long time to complete. I had to create many basic website features that we had to have. These include error pages, a search form, a forgot username/password form, a change name form, a change password form, and a delete account form. Creating error pages was odd, but once I learned how to use Django's built in functions for it, it was very easy. The search page was pretty basic: the user enters in a R number and whether or not the R group is archived (plus the year it was archived if it is) and it brings them to the page for that R group. The other forms were not so easy. For the forgot username/password form I had to figure out how to change the users password and then send them their password and username. The change name form was also mildly hard as I needed to let the user edit a model that I hadn't created, the User model, which is built into Django. The change password form was similar, but I had to verify the user knew their old password. Lastly, the delete account form had to have some sort of protection on it so the user didn't accidentally delete their account. I achieved this by giving them a random string they had to type in to delete their account.

```
def forgot(request):
    if request.method == 'POST':
        form = ForgotForm(request.POST)
        if form.is_valid():
            raw_password = ''.join(random.choices(string.ascii_letters, k=8)) + ''.join(random.choices(string.punctuation, k=2))
            user = User.objects.get(email = form.cleaned_data['email'], is_active = True)
            user.set_password(raw_password)
            user.save()
            yag = yagmail.SMTP('terrapiintrackercontact@gmail.com', oauth2_file = './oauth2_creds.json')
            yag.send(to = [form.cleaned_data['email']], subject = 'Terrapin Tracker Username and Password Retrieval', contents = '
            return redirect('login')
```

The last, and most arduous part of the project was editing/deleting turtles and measurements, along with proving the user a history of edits of each turtle and measurement. This task included more errors than I can count and was mainly a big logic puzzle. Most errors I encountered were not actual Python errors, but instead errors in my own logic. I had to include many verifications to ensure turtles wouldn't overlap with each other and measurements wouldn't overlap with each other. In the end I was able to achieve this goal. There is now a form for editing a turtle, deleting a

turtle, editing a measurement, deleting a measurement, and archiving many turtles or R groups at once.

III. Demo Video

Our demonstration video can be viewed from our GitHub README.md file, or the link below:

<https://youtu.be/l0yu8ONMzVQ>

IV. Reflection and Closing Thoughts

Lisa:

Through the course of this project, I encountered many issues with CSS, HTML, and Django. Learning how to implement Django into HTML and then use CSS to style the file took a ton of trial and error. Any time I had to style something with CSS, I would make the change, check the output on the website, and then continue making changes until I was satisfied with how it looked. The whole project itself was a learning experience; the Codecademy courses taught me the basics, but I did not fully understand HTML and CSS until I started creating pages for the website. There were so many things I didn't know at the beginning, like the importance of class names, grid formatting, and error lists. Looking back now, I think that our choice of a Python/Django backend and an HTML/CSS frontend worked out well. The languages worked so well together, and it made elements such as the pages, forms, and measurement data much easier to work with. I would definitely recommend Django to anyone looking to build a website next year. If I were to go back to the start of the semester with the knowledge I have now, I would definitely organize and correctly label every CSS class as well as create a new CSS file for every new HTML file. Our CSS pages were very messy by the end of the project: there were lines of code that were written to style a class, but we did not remember which element they styled. I would also take more advantage of the base CSS file. The base file extends to all HTML files so for elements like tables, white center boxes, and "go back" buttons, we ended up with a lot of copy-and-pasted code where we could have just used a general class in the base page.

For the Capstone class, I think the 2-3 week reports were perfect. I do not think that they took up too much class time; they were also useful for the midterm/final report in recalling what we did

at what time. If anything, maybe we could do more of a timeline than a paragraphed report, where in the last few minutes of every class students could bullet what they did that day, so it would be fresh in their minds. I also think that the first two weeks where we define our project and scope did cut into our coding time a bit, so I think that groups should have more of an idea and basic plan by the end of Data Science to reduce the thinking/planning stage of Capstone. As for the project topics, I think the guidelines we had worked: as long as students are learning something new they could create whatever they wanted, whether it be in creating a game or a website. Lastly, I would encourage students to work in groups next year: it helped so much with being able to bounce around ideas and fix each other's errors.

Angelo:

One of my biggest, consistent issues throughout the project was the learning curve of using all of the new tools. In spite of this, I would say that my biggest regret for the project was not starting the physical project sooner. I took several Codecademy courses in order to prepare for the project, but I feel that I did not fully grasp the concepts until I began to create HTML files and write CSS. While the courses I took were very comprehensive and did help me get a good foundation for the tools I used, I had to learn on my feet the dozen different ways to center a div and if it was better to use, set the display attribute to "grid" or "flex." As I became more familiar with HTML and CSS, these things came more naturally and what would have taken me several hours now only takes me twenty minutes. Keeping a consistent design on the website also helped me because it made it very easy to reuse the CSS I had already written with only a few modifications. The inspect feature on Google Chrome was essential to style Django forms, given that the HTML tags you would use to style it are hidden in the actual file.

In regards to the Capstone course itself, I feel like the current format works well. That being said, I feel that picking a project/team earlier during Data Science would help to speed up the process at the start of the semester. While I do think that all of the scope definition and project pitch paperwork was tedious and should be slightly condensed, ultimately I feel that it is necessary and should be done in future classes. In terms of progress reports, I feel that the once-every-three-weeks format worked very well, given that we were able to use the whiteboard as a to-do list.

My final suggestion for the course is that an "I wish I had known" section should be added to the progress report. This section would serve as a place where students can keep tabs on all of the things

that gave them trouble throughout the weeks and can be later compiled and organized by topics to help people in future classes.

Aidan:

Besides the many errors that every programmer encounters, I don't think anything went terribly wrong. Overall, our choice to use "the web framework for perfectionists with deadlines" was very smart. Django allowed us to achieve much more than we could have if we used a different, more complex backend. I am very satisfied with how our project turned out and I don't think I would have changed it at all if I went back to January.

It was absolutely interesting being the test group for this new class, so while the structure wasn't perfect, that's basically what I expected. The project report cadence of one every two weeks worked well I think, it kept us on track, although it was a bit frustrating to abandon the actual project at times.

We do plan to continue some development on this project over the summer, primarily in the form of uploading the project to the cloud most likely using an Amazon Web Services (AWS) server. We also need to figure out how to obtain a Uniform Resource Locator (URL). Hopefully this won't cost much, but we are prepared to spend a little money if needed.

For next year, I wouldn't make any huge, sweeping changes, but I would make some minor ones. For one, I wouldn't allow bigger groups, but I would highly recommend working in a group of three if possible. I think that having three people working on our project was critical to completing it on time. From what I can tell, some people had a project with similar scope to ours, but were working alone, and after completing the course I don't think these projects were super feasible to complete without making many concessions. I would also absolutely require students to pick their project during Data Science. It was frankly a waste of time spending the first few days of this course figuring out a project; I think it would've been much better if we could've jumped right in. I would not change the progress report format much at all, add a poster requirement (so many projects just wouldn't be done justice through a poster), and require the project to help society (I think allowing students full creative freedom is important). Lastly, I would absolutely share the Capstone Bible on day one. I wouldn't require them to update it weekly though. I think bi-weekly the same as the progress reports would work though.