



MATES Computer Science

Senior Capstone Project Bi-Weekly Progress Report

Project Title	Bit by Bit
Team Members	Anna Pitera
Dates Covered by Report	03/08/24 to 03/21/24
Link to Github	https://github.com/anna-pitera/Bit-by-Bit

1. Summary of Project

Overview:

Bit by Bit is a gamified to-do list and habit tracker website that encourages users to complete their daily tasks and maintain good habits by rewarding them with in-game currency, which can be used to purchase customization options and in-game benefits. The website is created with HTML, JavaScript, CSS, and Phaser, a JavaScript library for creating web games. The Django web framework is used along with PythonAnywhere, which allows programmers to host web apps online for free with restrictions that can be removed by purchasing a subscription. To use Bit by Bit, players must create an account with a username and password which stores their progress in a MySQL database, allowing them to close and reopen their progress at any point from any computer. Bit by Bit “resets” every day at midnight so users can check off their habits each day, and users are rewarded for “checking in” each day.

Website Design & Layout:

The design of the website is in a simple, easily reproducible pixel art style, and the main page of the website consists of five key features: the user’s character profile box, a to-do list layout, a store where users can spend their in-game currency on customization options and character accessories, a habit tracker display, and a few different number displays. The user’s character profile box displays the user’s customizable avatar/character and is made using Phaser sprite functionality. The to-do list is a list where users can enter and complete their daily tasks, and task completion rewards the user with currency. The store is a tiled display showing all obtainable items that can be purchased with currency, including character accessories and UI color themes. The habit tracker displays the user’s entered habits that they choose to maintain, along with a check box where users can check off whether or not they maintained their daily habits each day. The user’s habit “streak,”

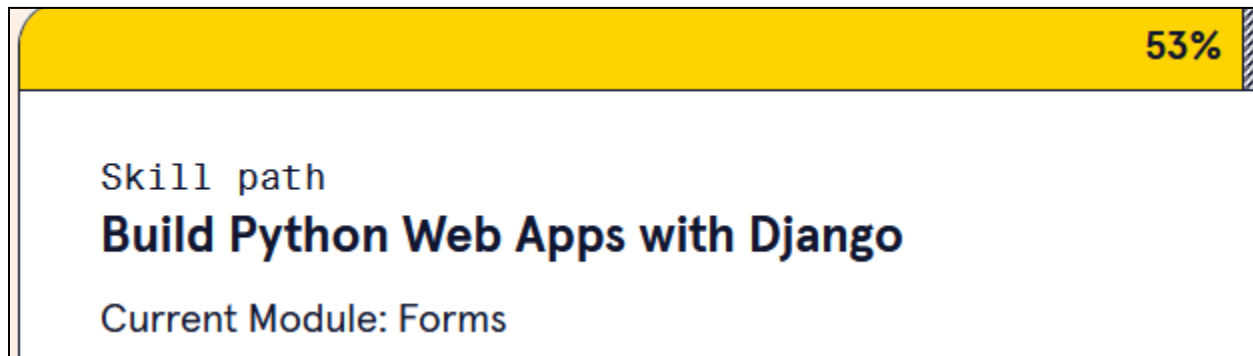
(the number of days they have maintained their desired habits in a row), currency amount, and total number of tasks on their to-do list are all shown in various number displays. Simple quality-of-life features are also included, such as the ability to undo the accidental completion of a task.

2. Summary of Progress this Period

Much of this progress period was spent figuring out Django functionality and tweaking it for my specific use cases. I already had the basis of the Django project set up through PythonAnywhere, but I also had to figure out how to make it work with VS Code and live HTML preview as well. I also worked on the Codecademy Django course, which taught me a lot about Django forms and databases. The Codecademy course also taught me more about the Django syntax, which is a little weird because you put it directly in the HTML code (but I'm learning it!). In terms of databases, I figured out how to connect to the PythonAnywhere database from MySQL Workbench on my home computer, and I also prepared to start storing some basic to-do list data in the database as well.

3. Detailed Progress this Period, separated by Team Member

I started this progress period by working on the Codecademy Django course. I didn't get that much done of it compared to the last progress period, but it was definitely meaningful and useful to do. I was at 35% complete at the end of the previous progress period, and now I'm at 53% complete.



After that, I came to the realization that I should probably figure out how to transfer the PythonAnywhere files to VS Code so I can start a little tiny bit more of the frontend to backend connection (with the databases and Django). So I zipped the PythonAnywhere mysite folder and put it into VS Code, and not surprisingly, nothing worked when I tried to run the Django development server, which is because of a few reasons:

1. The PythonAnywhere virtual environment was a bit different than the virtual environment I had created at the beginning of the semester.
2. The file paths (which Django uses a lot of) were all different on PythonAnywhere than what they were on my computer.
3. Django migrations, which are like "updates" you can post to your Django project sites, are stored in a couple of Python files, and they were a bit different on PythonAnywhere than the ones I had on my computer.

The first issue with the virtual environment was kind of difficult to figure out, mostly because I kept accidentally forgetting to activate the virtual environments in the console (or because I accidentally added the entire virtual environment folder to the staged changes with Git in VS Code...which ended up crashing VS Code a few times...).

The second issue with the file paths was easier to figure out; all I had to do was copy and paste some of the code, change the path in the pasted version, and comment out the PythonAnywhere version, see below.

```
# PythonAnywhere:
# STATIC_ROOT = '/home/annapitera/mysite/static'
# STATICFILES_DIRS = (
#     '/home/annapitera/mysite/static/node_modules',
#     '/home/annapitera/mysite/static/fonts/',
# )

# VS Code:
STATIC_ROOT = 'C:/Users/apitera/Documents/Capstone/Bit-by-Bit/mysite/static'
STATICFILES_DIRS = (
    'C:/Users/apitera/Documents/Capstone/Bit-by-Bit/mysite/static/fonts',
)
```

The third issue with the Django migrations was also pretty easy to fix; all I had to do was delete the old migration files and make new migrations. It's hard to explain, but "migrations" are kind of like versions of your Django project site? You have to re-migrate after bigger changes, especially changes to the important Django project files.

After that, I went on to the (in my opinion) much harder Django stuff. This included messing around with the settings.py file, trying to figure out how to run the Django development server through VS Code easier (spoiler alert: there's not really a way to do live preview with Django sites in VS Code, you just have to type a specific command sometimes to reload it, but other times it'll detect that you made changes and reload automatically, which isn't really live preview because you still have to use a browser to open it), and figuring out how to link views and models and urls.

I won't go into too much detail on views vs models vs urls, but it's basically just three different Python files that link stuff together. I wish I had screenshots of what those files looked like before this progress period, but this is what they look like now (not really much there now, but there will be more soon):

views.py (connects user requests to index.html):

```
mysite > bitbybit > views.py > index
1  from django.shortcuts import render
2
3  def index(request):
4      return render(request, 'index.html')
```

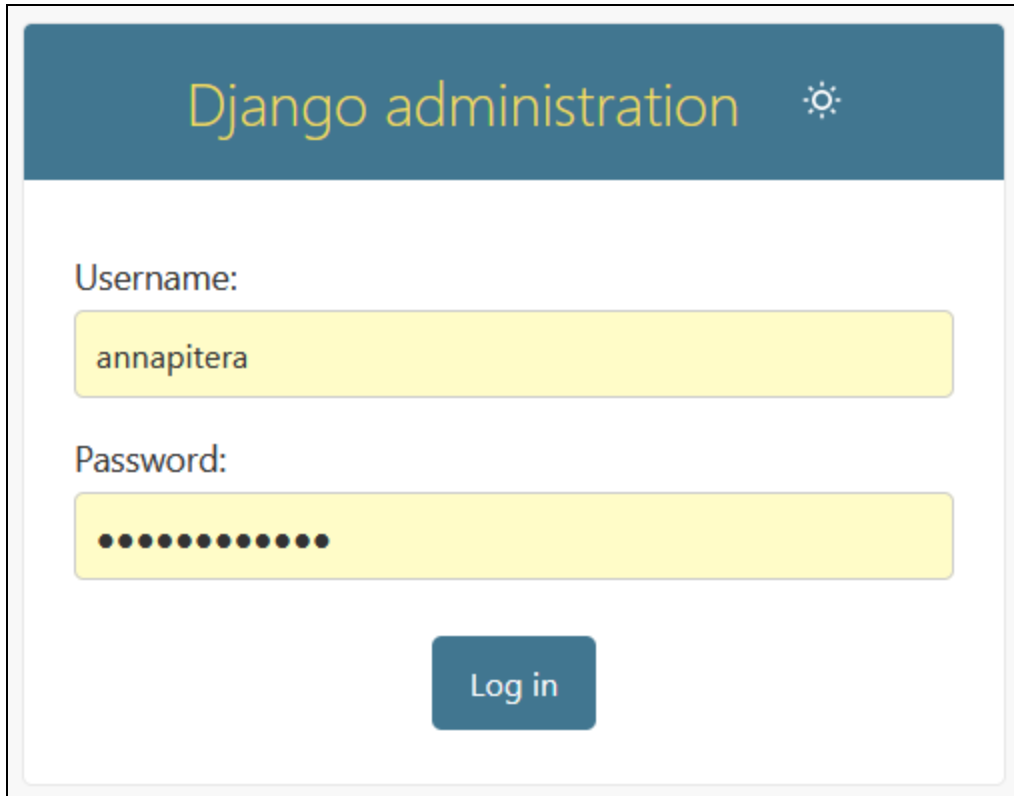
models.py (as of right now it contains the task model, which is going to be used to make the to-do list item entry form):

```
mysite > bitbybit > models.py > Task > __str__
1  from django.db import models
2
3  class Task(models.Model):
4      text = models.CharField(max_length=200)
5      is_done = models.BooleanField(default=False)
6      created_date = models.DateTimeField(auto_now_add=True)
7
8      def __str__(self):
9          return self.text
```

urls.py:

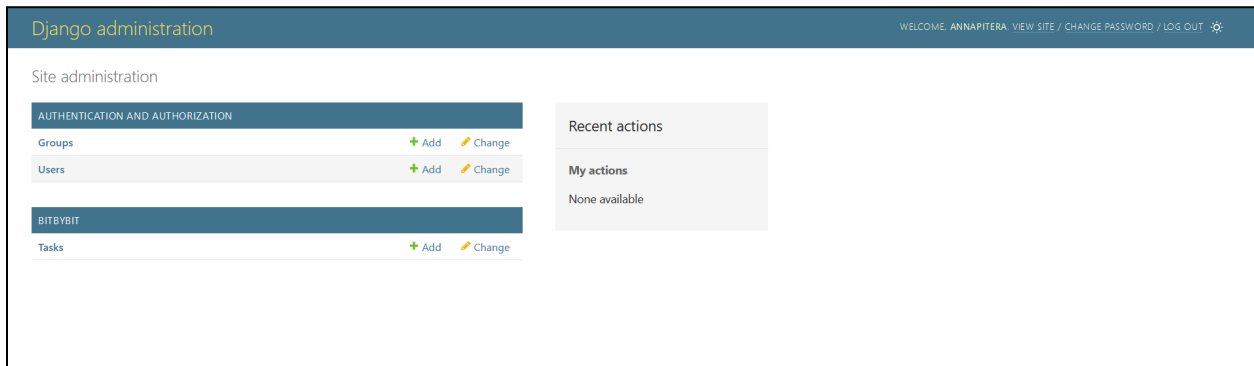
```
mysite > bitbybit > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('', views.index, name='index.html'),
6  ]
```

After a long time of figuring those files out, I then moved on to setting up the admin site. The Django admin site lets you manually add items to the database and see most of the database information. Here's what the login page for the Django admin site looks like (I obviously didn't make it look this good, it did that part automatically):



The image shows the Django administration login interface. At the top, there is a dark blue header with the text "Django administration" in yellow and a small gear icon. Below the header, the form has two input fields: "Username:" with the value "annapitera" and "Password:" with masked characters. A blue "Log in" button is centered below the password field.

And here's what it looks like once you're logged in:



The image shows the Django administration dashboard after logging in. The top header is dark blue with "Django administration" on the left and "WELCOME, ANNAPIERA VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. The main content area is white and divided into two columns. The left column, titled "Site administration", contains two sections: "AUTHENTICATION AND AUTHORIZATION" with links for "Groups" and "Users" (each with "Add" and "Change" links), and "BITBYBIT" with a link for "Tasks" (with "Add" and "Change" links). The right column contains a "Recent actions" section with a "My actions" link and the text "None available".

After doing that, I basically just researched some more database and forms stuff, so I found a couple of good guides that I plan on using in the next progress period.

4. Difficulties Encountered this Progress Period

- Django & PythonAnywhere
 - Again, a lot of this progress period was just trial and error; but this time it was mostly outside of PythonAnywhere because I already have that part set up to the point it needs to be at right now.

- I already mentioned the issues I encountered in the previous section (and I don't want to get too into the complicated details), but I basically figured out the issues mainly by Googling and using Stack Overflow but also by referencing the Terrapin Tracker repository. I have their repository downloaded, so I've just been opening up their settings.py and views.py and other files to make sure I'm doing things properly. It's really great to have that resource.

5. Updated Trello Board and Discussion

Link: <https://trello.com/b/Lzd1bk2c/bit-by-bit>

Changes:

- Added more links to "Resources" list
- Added a couple of tasks during this progress period involving Django that I ended up completing

6. Tasks to Be Worked on in Next Progress Period

- Continue Codecademy course
- Finish test form with to do list task entry
- Test code on PythonAnywhere

7. Additional Information

Django is so weird and cool. Also, research is over for the most part (2 more events in April and then we're done!), so I'll be able to work on this a bit more at home now.