MATES Computer Science

Senior Capstone Project Bi-Weekly Progress Report

Project Title	Bit by Bit
Team Members	Anna Pitera
Dates Covered by Report	04/19/24 to 05/02/24
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 JavaScript Canvas 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.

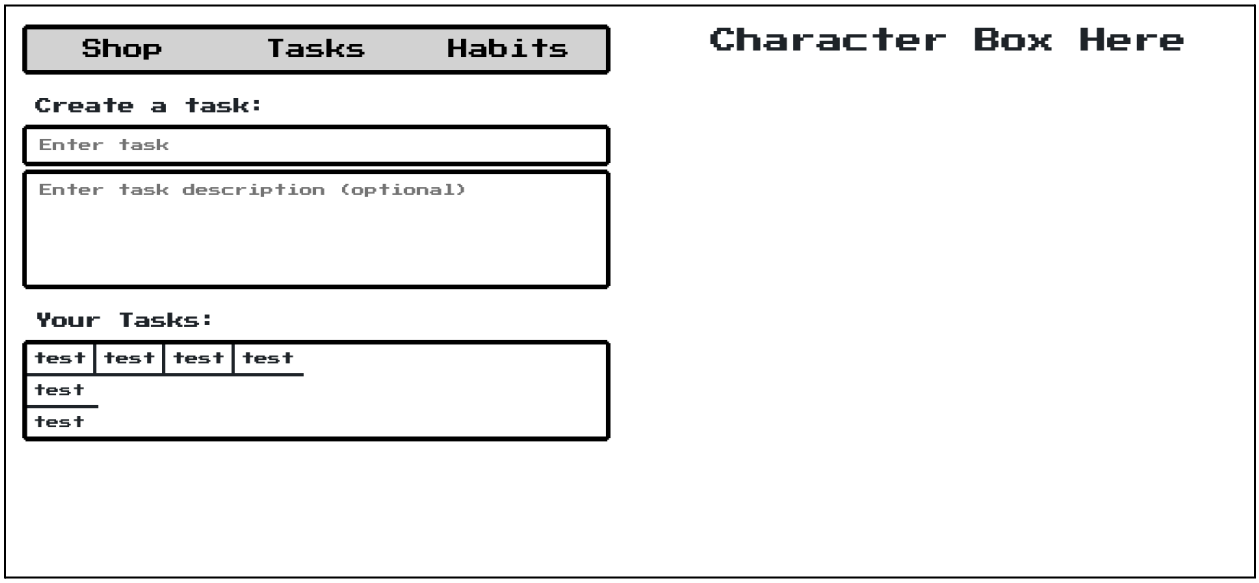
2. **Summary of Progress this Period**

This progress period was spent working on both the frontend and backend. In terms of frontend development, a lot of progress was made on determining the general layout of the site (split screen, navbar, navbar items/URLs, input boxes, character box, etc). Time was also spent stylizing the site to make it more visually appealing with the prebuilt NES.css pixel art style CSS library. Finally, the site was cleaned up so that everything lines up and scales properly. As for the backend, many issues were encountered with the Django models as a result of previous databases that weren't fully deleted getting in the way (i.e. past attempts to set up the database on VS Code were still there and were messing up the Django migrations). Luckily this was figured out along with the Django admin page, which allows superusers to view and modify items in the database.

3. **Detailed Progress this Period, separated by Team Member**

I decided to kind of "take a break" from updating the site on PythonAnywhere now that I already know how to do it, so I've been using VS Code and a local throwaway database to test Django and frontend stuff without having to constantly update the site. That means that the PythonAnywhere site is technically still broken, but I have everything fixed locally, so I should be able to update the live site on PythonAnywhere once all the development in VS Code is complete.

I worked a lot on the frontend this progress period. At the end of the previous progress period, I had basically nothing to show for the frontend; now that's changed! Here's what the site looks like now:



The navbar at the top of the left side of the screen took a while to figure out, mostly because I decided to use the NES.css pixelated library I found in the beginning of the semester again (and it took a while to figure out how to remove some of the NES.css borders and styles and replace them with my own). While it did take a while, I think using the NES.css library will make things a lot easier because it has built-in pixelated site elements (even if I have to manually change them myself once in a while). The navbar also links to different URLs, but they're not set up yet.

The “Create a task” section of the site is looking pretty good as well. I’ll have to add in the calendar picker widget for the due date column of the database, but I’ll do that in the next progress period once I get a task submission button working. The “Your tasks” section of the site is pretty temporary for right now (that’s why it has a table instead of a list), but overall I think the site looks good. It took a while to get things aligned with the right margins and padding too. Also keep in mind that the screen is still split with a scroll bar on the left screen, you just can’t see it because there’s nothing that extends that far down yet. Another thing that you can’t tell from the screenshot is that the NES.css library makes the cursor on the website into a pixelated hand pointing cursor, which I think is really cool.

Here’s the code for tasks.html, which is the page that opens automatically when you go to the localhost site:

vscodemysite > bitbybit > templates > tasks.html

```
{% block head %}
<head>
    <title>Bit by Bit</title>
    <link rel="stylesheet" href="{% static './nes.css/css/nes.min.css' %}">
    <link rel="stylesheet" href="{% static './style.css' %}">
    <style>
        html, body, pre, code, kbd, samp {
            font-family: "Press Start 2P";
            src: url("./2p.ttf");
        }
    </style>
</head>

<body>
    {# Left side of split screen #}
    <div class='split left'>
        <div class="topnav pixel-corners pixel-corners--wrapper nes-btn">
            <div class="topnav-left">
                <a href="#shop">Shop</a>
            </div>
            <div class="topnav-centered">
                <a href="#tasks">Tasks</a>
            </div>
            <div class="topnav-right">
                <a href="#habits">Habits</a>
            </div>
        </div>
        <h1 class="titles nes-text">Create a task:</h1>
        <div class="nes-field a">
            <input type="text" id="name_field" class="input nes-input pixel-corners" placeholder="Enter task" required>
            <textarea id="name_field" class="input nes-input pixel-corners" placeholder="Enter task description (optional)" rows="5" cols="30"></textarea>
        </div>
        <h1 class="titles nes-text">Your Tasks:</h1>
        <div class="pixel-corners nes-table-responsive">···
        </div>
    </div>

    {# Right side of split screen #}
    <div class='split right'>
        <h1>Character Box Here</h1>
    </div>
</body>
{%endblock%}
```

And here’s the code for style.css (excluding the pixel borders class because it’s really long):

```css
h1 { text-align: center }

.tables {
  align-items: center;
  display: inline-block;
  justify-content: center;
  flex-grow: 1;
}

.titles {
  font-size: 20px;
  text-align: left;
  padding-left: 1em;
  padding-top: 1em;       You, 3 days ago • Figured out how to align everything
}

.input {
  width: 45.25em;
  resize: none;
}

.topnav {
  background-color: lightgray;
  overflow: hidden;
  font-size: 25px;
  display: flex;
  justify-content: space-between; /* Adjust space between items */
  align-items: center;
}

.topnav a {
  color: black;
  padding: 0.25em;
  text-align: center;
  display: block; /* Make the anchor element a block-level element */
  width: 9.5em; /* Set a fixed width for each item */

  text-decoration: none;
}

.topnav a:hover {
  background-color: darkgray;
  color: black;
}
```

```css
.topnav-centered {
  display: inline-block;
  justify-content: center;
  align-items: center; /* Align items vertically */
  flex: 1; /* Take up remaining space */
}

.topnav-right {
  font-size: 25px;
  align-items: center;
  flex: 1;
}

.split {
  height: 100%;
  width: 50%;
  position: fixed;
  z-index: 1;
  top: 0;
  overflow-x: hidden;
  padding-top: 20px;
}

/* Control the left side */
.left {
  left: 0;
  padding-left: 1em;
}

/* Control the right side */
.right {
  right: 0;
  padding-left: 1em;
}
```

In terms of the backend, I set up the basic parts of the Tasks Django model, so each task in the database now has three parts: task_title, task_desc (which is an optional field), and time (which records the time the task was created). I'm also going to add a due_date part and a boolean is_done part to the Tasks model in the future.

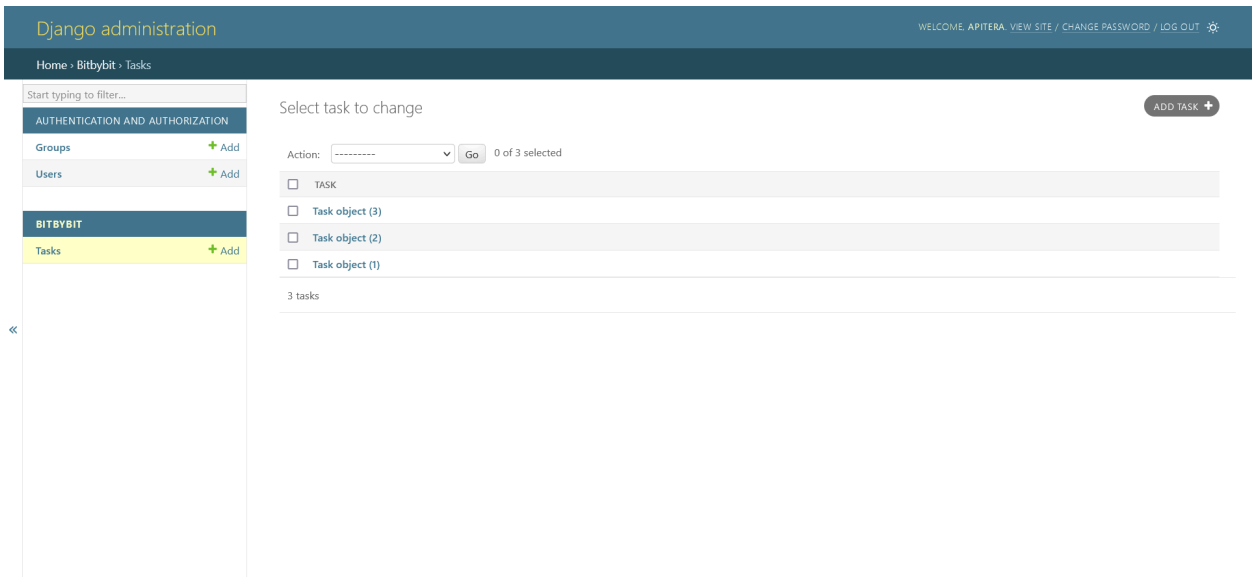
This is what models.py looks like:

```
anna-pitera, 2 days ago | 1 author (anna-pitera)
from django.db import models

anna-pitera, 2 days ago | 1 author (anna-pitera)
class Task(models.Model):
    task_title = models.CharField(max_length=70)
    task_desc = models.TextField()
    time = models.DateTimeField(auto_now_add=True)
```

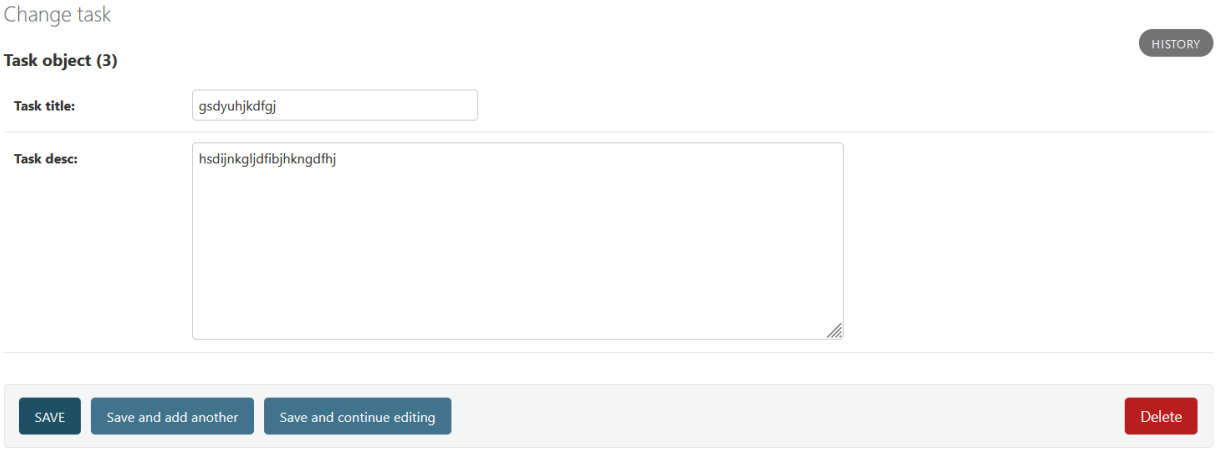
And then I registered the model with the admin site, so when I'm logged into the admin site I can add, view, edit, and delete tasks. Here's admin.py:

```
from django.contrib import admin
from bitbybit.models import Task

admin.site.register(Task)
```

And here's what the tasks look like on the admin site:



And then when you click on a task, it shows each part (excluding time because that's not fully implemented yet);



4. **Difficulties Encountered this Progress Period**
   - Django
     - As always, Django is a little difficult.
     - I was having trouble with making and implementing migrations in the Django project because I kept getting errors saying that I already had migrations made, therefore it wouldn't let me make new ones and synchronize the database. It was a whole issue that's kind of hard to explain.
     - I eventually figured out that, when I was first testing different methods of creating the Task model a few weeks ago and deleted the test models, they didn't delete all the way. Basically, there were a bunch of SQL tables left over from the testing stage, and some of them had the same names as the new tables I was creating.
     - I fixed this by dropping every table from the database, effectively resetting the whole database and making it completely empty. It turns out that you can do this with one single Django command (thanks Stack Overflow!), so that was cool. I added that command and Stack Overflow post as a reference into the Lessons Learned document too, just in case someone ever has the same issue.
   - HTML/CSS:
     - Figuring out how to line everything up and make everything the same size took an embarrassingly long amount of time. Things had extra padding and margins, and I had no idea where they were coming from.
     - I finally thought to check the NES.css library's (really long) CSS file to see if anything was quietly breaking stuff through that. Thankfully, it was only a couple lines of code that I had to change/remove to align everything.
     - Another annoying issue I realized I had was that the Inspect Element panel kept moving stuff around when I wanted to test new HTML/CSS code with it. I learned that this was because every element in the site was sized dynamically with the browser window, so once I figured out how to stop it from doing that, everything started working the way I wanted it to. I put this in the Lessons Learned doc as well.

5. **Updated Trello Board and Discussion**

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

Changes:

- Added more links to “Resources” list
- Added a few tasks of varying priority to do
- Completed a few tasks

6. **Tasks to Be Worked on in Next Progress Period**
    - Frontend
        - Add submit button so the user can actually interact with the database and submit task information.
        - If time permits, add calendar date picker UI.
    - Backend
        - Finish adding full CRUD functionality to task form (i.e. make a functional submit button).
        - If time permits, add due dates to the database.

7. **Additional Information**

Things are working!! I’m a little worried that I won’t be able to implement all of the things I wanted for the site by the end of the semester, but at least I’ll have something to show.