



MATES Computer Science

## Senior Capstone Project Bi-Weekly Progress Report

Project Title	MorningBread
Team Members	Max Bradshaw, Jake Dorick
Dates Covered by Report	April 22nd - May 3rd, 2024
Link to Github	<a href="https://github.com/maxbshaw17/MorningBread">https://github.com/maxbshaw17/MorningBread</a>

1. **Summary of Project** (Provide a one paragraph summary of your project. You can largely copy/paste this from one progress report to the next, unless there are significant changes.)

To provide a personalized morning financial report to our users by utilizing a web scraping tool for financial website articles to find the general topics of the articles. In doing so, we are creating and designing a website to give the public a general grasp of financial news for the day, over the course of the day with hourly, highly summarized news headlines/stock tickers. An account system will allow users to follow certain companies and manage tickers.

2. **Summary of Progress this Period** (Provide a high-level, one-paragraph overview of what was accomplished this progress period collectively by the team.)

### Jake

Two main goals were the priority of the past two weeks. The first goal was to finalize the dynamic articles on the website with the completed database. The database was not completely finalized until a few days before writing this, so it was put on the back burner for the first week of this sprint. The other goal was introducing an account system into the website, utilizing the profile and signup HTML pages I created a while ago. It is, as I expected, a daunting task, but I am confident that it will be completed by the end of the project and its functionality will be implemented. In the end, dynamic articles were completed as Max finished the backend portion of the tasks.

---

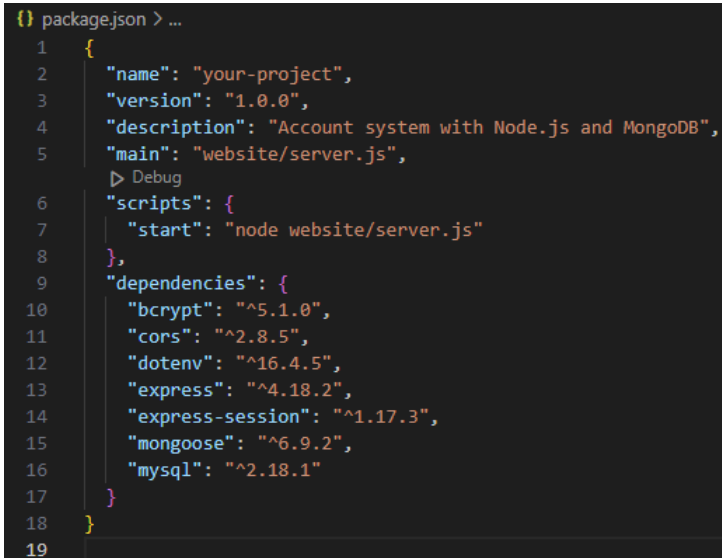
## Max

During week one, I worked on refactoring the ML functions to be more generalized, as well as completely rewriting the chatGPT summarization function. Week two was spent working on the VPS server. I decided to go with a Digital Ocean Droplet, as it ran ubuntu and is fairly powerful for just \$4 per month. On the server, I automated the summarization script execution costing me ~\$0.14 per day in GPT tokens. I am now currently working on actually hosting the site on the same server.

### 3. Detailed Progress this Period, separated by Team Member (Provide detailed information on the progress that you made in the reporting weeks. Include screenshots of code, your game or website, etc. Each team member should have a separate subsection covering their accomplishments. Not including screenshots, this section should be 1-2 pages.)

## Jake

- This sprint was a section of introducing a lot of new things to me, especially in the realm of hosting JavaScript files
- Node.js had to be hosted to properly maintain the API that connects the articles in the MySQL database to the website, and the newly used MongoDB had to have a separate localhost server connection in my attempts to start creating an account system
- With this, Node.js's npm package manager was used to install all of the packages needed for Node.js and MongoDB
- In doing so, it created a JSON file to fulfill all the major packages needed to introduce the systems implemented
- It also created a node\_modules folder in the project which includes thousands of JSON files, licenses, readmes, and, most importantly, JavaScript files



```
{
  "name": "your-project",
  "version": "1.0.0",
  "description": "Account system with Node.js and MongoDB",
  "main": "website/server.js",
  "scripts": {
    "start": "node website/server.js"
  },
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "mongoose": "^6.9.2",
    "mysql": "^2.18.1"
  }
}
```

- Moving onto other progress, I started with the account system (since I was still waiting on the completed database before moving on to finishing the articles)
  - This process is extremely frustrating, reading console logs and error messages one by one to make small progress is going to be the route for this process
  - Overall, I stopped at a pretty good point
-

- First, I had to establish a server with MongoDB to stash the accounts coming in through the HTML local host

```
const express = require('express');
const app = express();
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const session = require('express-session');
const path = require('path');
const cors = require('cors');

// Middleware
app.use(cors());

// Connect to MongoDB
mongoose.connect('mongodb://localhost/myapp', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log('MongoDB connected'))
.catch(err => console.log(err));

// User schema
const userSchema = new mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  password: String,
});

// User model
const User = mongoose.model('User', userSchema);

// Middleware
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(session({
  secret: 'your-secret-key',
  resave: false,
  saveUninitialized: true,
}));
app.use(express.static(path.join(__dirname, 'website')));

// Sign up route
app.get('/signup', (req, res) => {
  res.sendFile(path.join(__dirname, 'website/signup.html'));
});

// Login route
app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, 'website/profile.html'));
});

// Signup POST route
app.post('/signup', async (req, res) => {
  const { firstName, lastName, email, password } = req.body;

  try {
    // Check if user already exists
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: 'User already exists' });
    }

    // Hash password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Create new user
    const newUser = new User({
      firstName,
      lastName,
      email,
      password: hashedPassword,
    });

    // Save user to database
    await newUser.save();

    res.status(201).json({ message: 'User created successfully' });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Internal server error' });
  }
});

// Login POST route
app.post('/login', async (req, res) => {
  const { email, password } = req.body;

  try {
    // Find user by email
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(400).json({ message: 'Invalid email or password' });
    }

    // Compare password
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      return res.status(400).json({ message: 'Invalid email or password' });
    }

    // Set user session
    req.session.user = user;

    res.status(200).json({ message: 'Login successful' });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Internal server error' });
  }
});

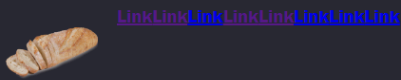
// Start the server
app.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

- I know the code above is a lot and it's pretty small but in a nutshell what it is doing is:
  - A) Connecting to a MongoDB server;
  - B) Determining whether it is a login route or signup route;
  - C) Creating a POST request with the server to either find a user or create a new one; and
  - D) Compare passwords (login route) or hash passwords (signup route)

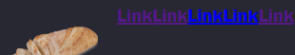
```
43 try {
44     const response = await fetch('/login', {
45         method: 'POST',
46         headers: {
47             'Content-Type': 'application/json',
48         },
49         body: JSON.stringify({ email, password }),
50     });
51
52     const data = await response.json();
53     console.log(data.message);
54     // Handle success or error
55 } catch (err) {
56     console.error(err);
57 }
58 });
```

- A website pop-up occurs if the user does not implement a matching password
- More pop-ups are to come for other issues like incorrect passwords or usernames not existing
- Finally, the bread and butter of the project, the articles

**Stock buybacks are on the rise, while FTX customers are expected to recover lost funds and Uber and Shopify stocks face earnings-related challenges. (8)**



**JPMorgan and Nomura reduce exposure to Segantii in response to insider trading cases. (5)**



#### Almost Final Product ^

- Aesthetically-wise, I fixed having no spacing between the articles (weird bug) and I removed the "Show Summary" button function
  - I was under the impression that there were headlines and summaries but that's wrong, the headline is the summary
- Functionally-wise, I had to fetch from two separate databases to get the newly implemented functions
  1. Links to the original articles
  2. Magnitude based on the frequency of articles with similar headlines appearing on the internet (I assume Max will discuss this further)
  3. Not including repeating article headlines

```
c.execute("SELECT summarized_articles.summarized_headline,
articles_grouped.link, summarized_articles.magnitude FROM
summarized_articles, articles_grouped WHERE
summarized_articles.group_id = articles_grouped.group_id ORDER BY
summarized_articles.magnitude DESC")
```

This is what grabs from the MySQL database ^

---

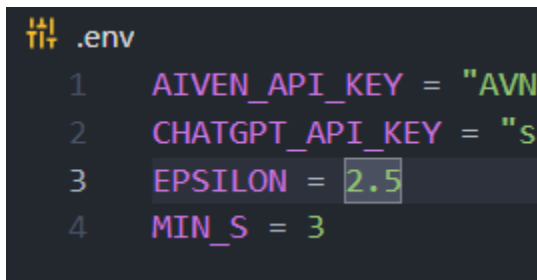
- A few more functions were added to articles.js to not include repeating article headlines, nothing too heavy though

```
articles.forEach((article, index) => {  
  // Check if the article has already been displayed  
  if (displayedArticles.has(article.summarized_headline)) {  
    return; // Skip this article  
  }  
  
  displayedArticles.add(article.summarized_headline); // Add the article to the set  
  
  const featuredArticleElement = document.createElement('div');  
  featuredArticleElement.classList.add('featured-article');  
  if (index === 0) {  
    featuredArticleElement.classList.add('first-article');  
  }  
  featuredArticleElement.innerHTML = `  
    <h1>${article.summarized_headline}</h1>  
    <div class="article-preview">  
        
      <div>  
        <a href="${article.link}" class="article-link" target="_blank">Original Article</a>  
      </div>  
    </div>  
  `;  
};
```

- *Extra I forgot to mention earlier:* Developed and finished the search bar's functionality (JavaScript) to look for keywords in the articles (highlights words you search for)

## Max

- Added my API keys to a .env file so as not to upload them to github, as well as some variables for the grouping algorithm so I can easily tweak in vim on the server



```
1  AIVEN_API_KEY = "AVN  
2  CHATGPT_API_KEY = "s  
3  EPSILON = 2.5  
4  MIN_S = 3
```

- Rewrote the text cleaning functions to be much easier to read, and fixed the lemmatization algorithm so words are properly tagged with their part of speech and lemmatized accordingly
-

```

def clean_sentence(sentence: str):
    def preprocess_text(text):
        # convert text to lowercase
        text = text.lower()

        # remove special chars and digits using regex
        text = re.sub(r'\d+', '', text) # remove digits
        text = re.sub(r'[^\w\s]', '', text) # remove all special characters

        # tokenize text
        tokens = nltk.word_tokenize(text)

        return tokens

    def remove_stopwords(tokens):
        stop_words = set(stopwords.words('english'))
        filtered_tokens = [word for word in tokens if word not in stop_words]

        return filtered_tokens

    def get_wordnet_pos(treebank_tag):
        # converts treebank tagging convention to wordnet tagging convention
        if treebank_tag.startswith('J'):
            return wordnet.ADJ
        elif treebank_tag.startswith('V'):
            return wordnet.VERB
        elif treebank_tag.startswith('N'):
            return wordnet.NOUN
        elif treebank_tag.startswith('R'):
            return wordnet.ADV
        else:
            return ''

    def perform_lemmatization(tokens):
        lemmatizer = nltk.stem.WordNetLemmatizer()

        treebank_pos_tags = pos_tag(tokens)
        wordnet_pos_tags = [(tag_set[0], get_wordnet_pos(tag_set[1]))
                             for tag_set in treebank_pos_tags]
        # lemmatizes with part of speech tag if provided, and without if not
        lemmatized_tokens = [(lemmatizer.lemmatize(tag_set[0], tag_set[1]) if tag_set[1] != ''
                              else lemmatizer.lemmatize(tag_set[0]))
                              for tag_set in wordnet_pos_tags]

        return lemmatized_tokens

```

- Rewrote chatGPT prompt function to accept and return a dataframe

```

def summarize_sents(prompt_message, API_KEY):
    client = OpenAI(api_key=API_KEY)

    completion = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a text summarizer. You are given multiple sentences and you must summarize them into a single sentence."},
            {"role": "user", "content": prompt_message}
        ]
    )

    return completion.choices[0].message.content

def summarize_group_df(df: pd.DataFrame, API_KEY) -> pd.DataFrame:
    # group by group_id and create list of groups
    grouped_df = df.groupby('group_id')
    groups = list(grouped_df.groups)

    grouped_sent_lists = [] # list of tuples (group, len([sents]), [sents])

    for group in groups:
        if group >= 0: # filter out ungrouped (-1)
            grouped_sents = []

            for headline in grouped_df.get_group(group)['headline']:
                grouped_sents.append(headline)

            grouped_sent_lists.append((group, len(grouped_sents), grouped_sents))

    summarized_sents = []
    for sent_group in grouped_sent_lists:
        sents = sent_group[2]

        summary = summarize_sents(f"{sents}", API_KEY)

        summarized_sents.append((sent_group[0], sent_group[1], summary))

    return pd.DataFrame(summarized_sents, columns = ['group_id', 'magnitude', 'summarized_headline'])

```

- Rewrote the join groups method to work with duplicate rows; SQL is shoddy but works

```
def join_groups(self):
    insert_sql = """
    INSERT INTO articles_grouped (group_id, headline, link, date, source)
    SELECT headline_groups.group_id, articles.headline, articles.link, articles.date, articles.source
    FROM (SELECT articles.*, row_number() OVER (ORDER BY id) AS seqnum
          FROM articles
          ) articles JOIN
    (SELECT headline_groups.*, row_number() OVER (ORDER BY id) AS seqnum
    FROM headline_groups
    ) headline_groups
    ON articles.seqnum = headline_groups.seqnum;"""

    try:
        self.mycursor.execute(insert_sql)
        self.articles_db.commit()
        print(f'inserted {self.mycursor.rowcount} rows to "articles_grouped", {self.get_row_count("articles_grouped")} total rows')
    except Exception as error:
        print(f"error joining tables: {error}")
        self.articles_db.rollback()
    finally:
        return self.mycursor.rowcount
```

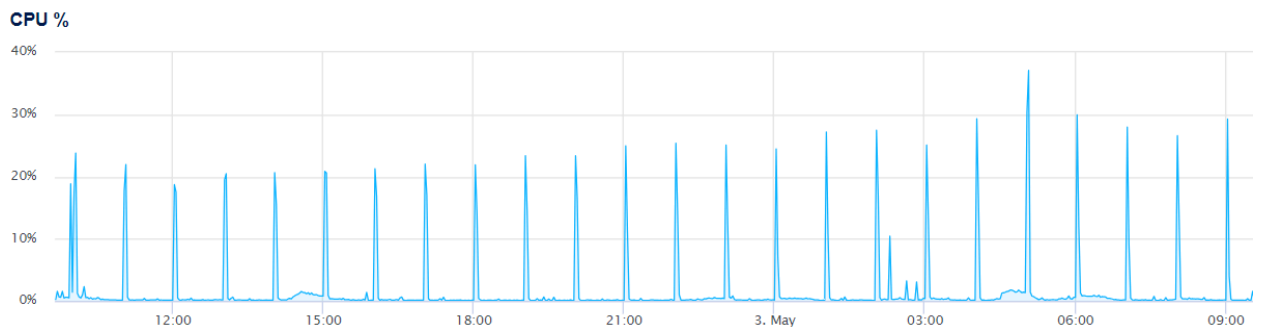
- Added a runtime feature to the script so I can see the total runtime of each section for debugging purposes

```
scraped 90 articles from finviz in 1.582 seconds

DevTools listening on ws://127.0.0.1:56594/devtools/browser/4f99062e-47be-4d21-9e40-af5d2db96dc4
[0503/092913.890:ERROR:ssl_client_socket_impl.cc(879)] handshake failed; returned -1, SSL error code 1, net_error -100
[0503/092913.958:ERROR:ssl_client_socket_impl.cc(879)] handshake failed; returned -1, SSL error code 1, net_error -100
scraped 50 articles from yahoo in 19.12 seconds
scraped 60 articles from marketwatch in 1.185 seconds
error inserting into "articles": 1467 (HY000): Failed to read auto-increment value from storage engine
deleted 0 rows older than 1.5 days from "articles", 900 total rows
deleted 0 duplicate rows from "articles", 900 total rows
deleted all (900) rows from "articles_grouped"
deleted all (900) rows from "headline_groups"
inserted 900 rows to "headline_groups", 900 total rows
inserted 900 rows to "articles_grouped", 900 total rows
deleted all (41) rows from "summarized_articles"
inserted 41 rows to "summarized_articles", 41 total rows
finished running in 91.131 seconds
PS Z:\MorningBread>
```

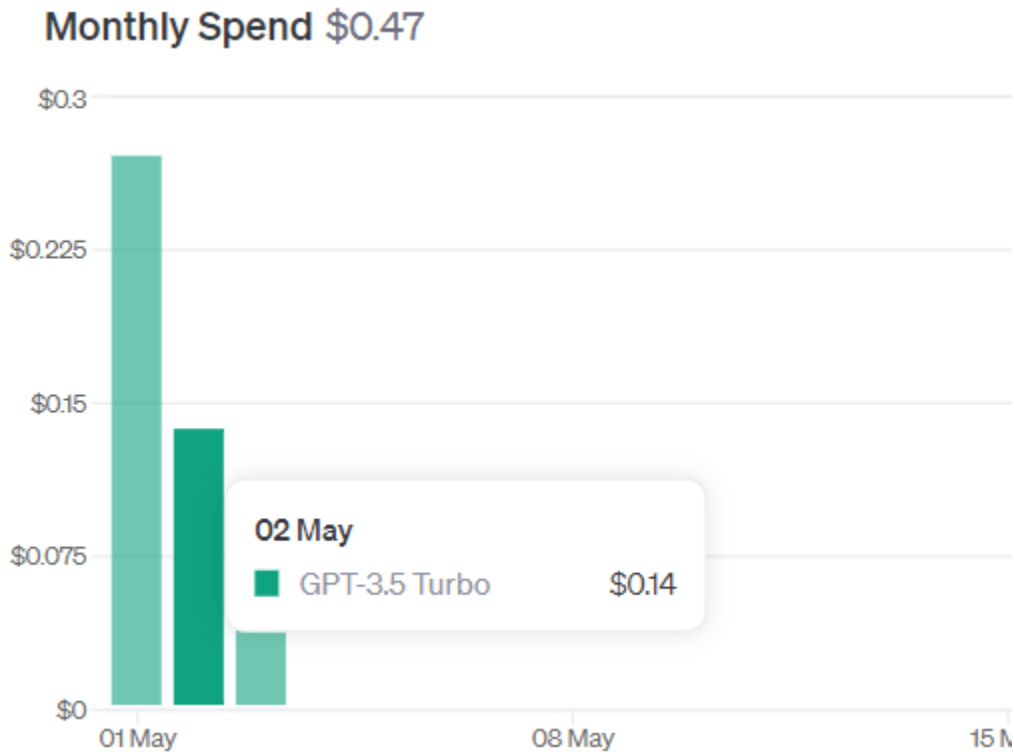
- Created Droplet and used CRON to automatically execute script

```
#
# m h dom mon dow   command
0 * * * * /root/MorningBread/article_api/venv/bin/python3 /root/MorningBread/article_api/sql_run.py
```





- ChatGPT prices - At first I ran the script every half hour but it seemed unnecessary so I changed to every hour to save money



- Purchased the domain morningbread.lol
- Started nginx server on the VPS and connected the domain
  - Website shows default nginx landing page right now
  - Have yet to implement SSL certification so the website is currently blocked by MATES firewall

4. **Difficulties Encountered this Progress Period** (Provide detailed information on the difficulties and issues that you encountered in the reporting weeks. Discuss mitigation strategies for how you got around or plan to get around these issues.)

**Jake**

- Issues with server-hosing on multiple localhosts
    - It took me a while to understand how to even run the MongoDB server, but I figured out that by using the npm manager from Node.js, it will run the server from terminal commands
  - JavaScript has been giving me so many problems, it has so much functionality but it often makes it impossible to understand
-

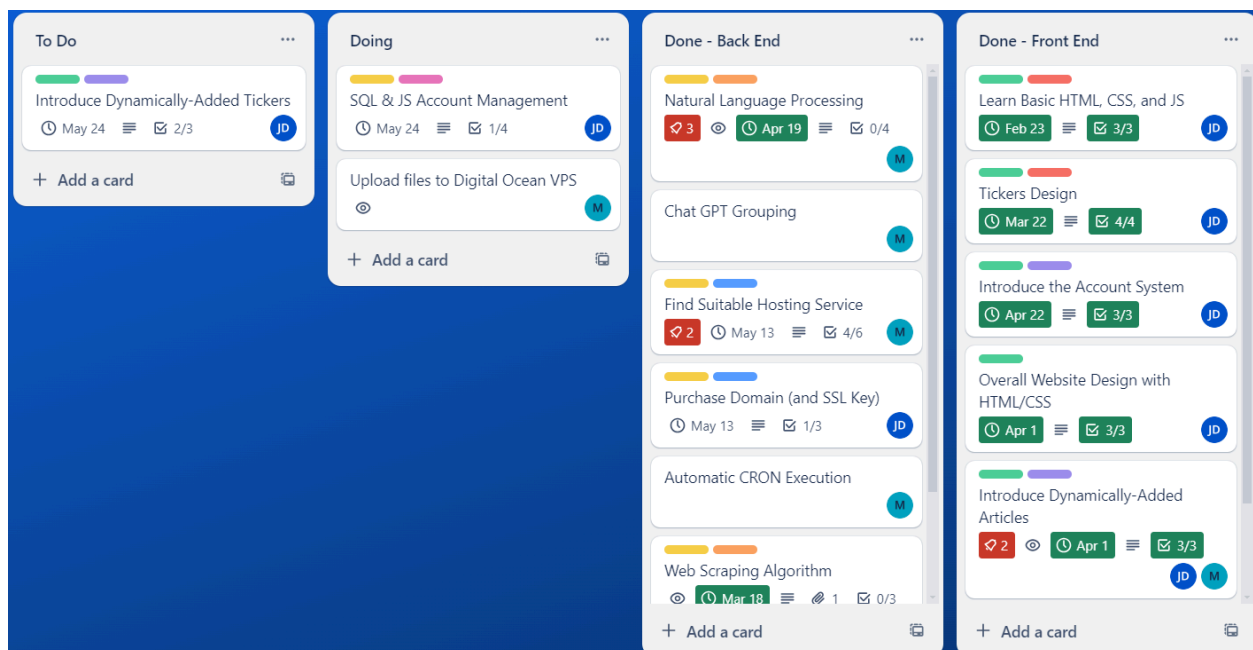
- A lot of console logs have helped guide me through troubleshooting
- Git won't allow you to show passwords in commits (took me a day to realize this) and then I had to learn how to utilize a .env file (still clearly don't since I didn't save my .env credentials anywhere except my Capstone laptop)

## Max

- SQL was difficult when I had to merge two tables with duplicate entries, eventually figured it out
- I'm still learning to navigate with the terminal
  - Vim is confusing
  - I had to install pipreqs to install all of my packages, but the generated requirements.txt had several errors that I had to manually fix
  - Installing chrome with no GUI is a headache, and it also doesn't like to run as root so I had to find a workaround for that
- SSH is amazing once it's set up but terrible before that

## 5. Updated Trello Board and Discussion (Provide screenshot of and link to updated Trello board. Discuss any changes made to board since last progress report and why.)

<https://trello.com/b/sSuMPdYn/morningbread>



Little changes have been made other than introducing new large tasks into "Doing" and "Done." Separation of back-end and front-end finished tasks was made to prevent one list from overpowering the Trello board. Domain/Hosting services are finally being worked on.

6. **Tasks to Be Worked on in Next Progress Period** (Discuss the tasks to be worked on in the following two weeks. Discuss who is working on each.)

**Jake**

- Either Max or I will continue server hosting on an actual domain and ensure all of the functionality of our scripts remain the same as if we were running on localhost ports
- I plan, now that the articles are done, to work heavily on the account system and get that working
- I want to implement some sort of personability to articles based on searches, but I don't know how realistic this is with the amount of time we have left

**Max**

- Hosting with nginx
  - Need to restructure the website's files to be in a single folder so nginx can handle it
- Implement SSL certification so I can actually go on the website

7. **Additional Information** (Provide any additional information that you want to provide in this section; for example, one of your teammates is going away next week, your Github account is gone, etc. It could be good news as well.)

**Jake**

Nothing major to report.

**Max**

Nothing here

---