Kettering University Social Space

Kalil Black, Domenico Alfano, Joshua Woods

Department of Computer Science, Kettering University

blac5369@kettering.edu, alfa7198@kettering.edu, wood2944@kettering.edu

***Abstract*-This report covers the project that was completed in CS-351 which entails an image upload form for the students of Kettering that allows for the sharing of social events on campus. It also covers the methods used to achieve this, involving Node.js and AWS as a web server. It also covers our initial goal, challenges we faced, and the in-depth technical specifications that were used to complete this project.**

Keywords- AWS - Amazon Web Services; Python; Node.js; Express; EC2 - Elastic Compute Cloud;

I. INTRODUCTION

Our product is a website that is available to the students of Kettering and has a goal of bringing the students of Kettering together. You will be able to upload a picture of your choice, with some text, and it will then appear on the site displaying from most recent to oldest. This site will be linked to from the MyKettering page for easy access for the students of Kettering. We drew similarities from other sites that are similar such as Instagram, but differ in our smaller user base and differing goals. Our site is simple, yet serves its purpose, it has four input fields, a title field, a name field, a body field, and a file select field to choose which image you would like to upload. Once the form is filled out, the user clicks submit, and the information is submitted to the database. The latest post then appears on the top of the screen displaying all of the information entered in the field.

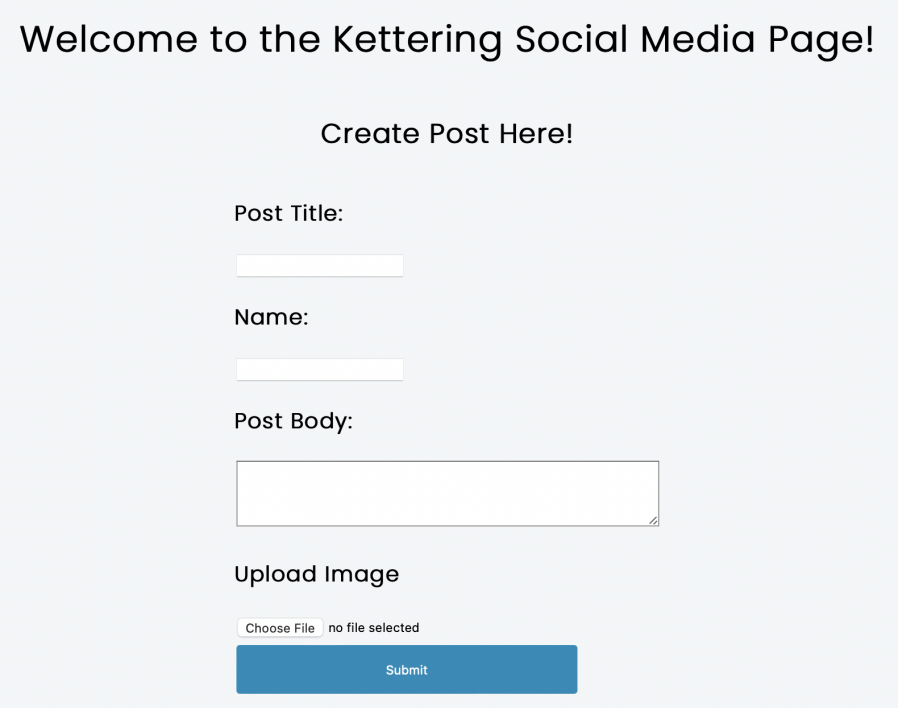


Figure 1: Posting Form

II. OUR MOTIVATION

Our motivation for this project was to provide a space for students to share events on campus that serves the purpose of bringing the campus together. As our site is similar to ones like Instagram, we hope to be more localized so that students can easily access and view other students' posts, and do not need to worry about having to follow anyone, or anything of that nature. With an easily accessible location for events to be posted about, everyone on campus can enjoy the activities that are going on and around campus and enjoy the full college experience. Students who are more connected with each other will have a better time while they are at school and will have more access to events that are happening around Kettering. Seeing images of what your fellow students are doing on campus will allow for you to feel more connected with the people around you, and provide the opportunity to attend more events that are on campus. The end goal is that all of the students know their fellow students and the overall mental health of students is higher, and that the experience while they are on campus is the best it can be.

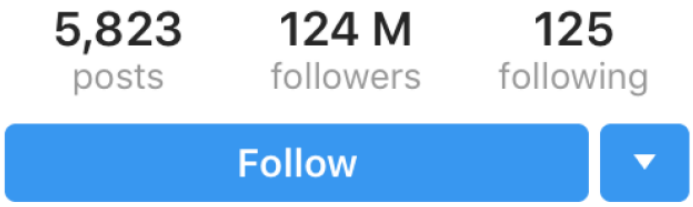


Figure 2: Kylie Jenner follower count

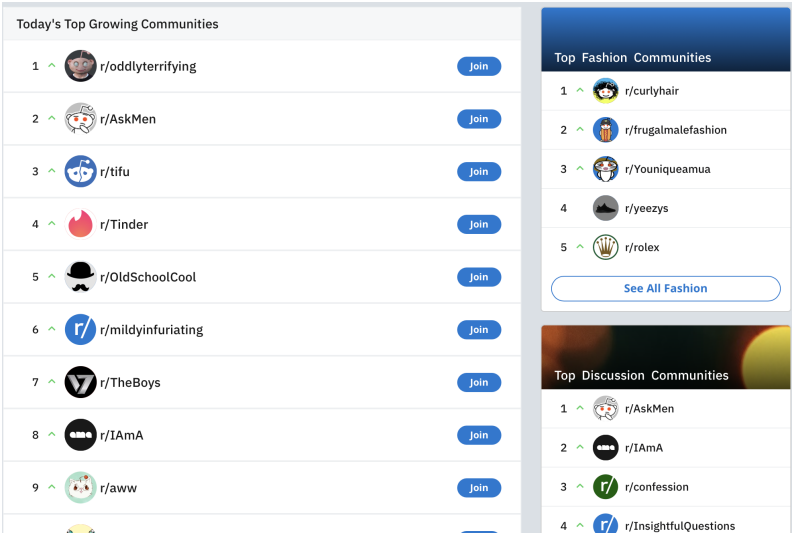


Figure 3: A fraction of the subreddits on Reddit

III. METHODOLOGY

At its core, this project was to become a cloud hosted full-stack web application. The backend was written using NodeJS, Express, Express-Handlebars, Express-FileUpload, and MySQL. The MySQL database was hosted using Heroku, a Platform as a Service provider that can host databases. The rest of the application was hosted on an AWS EC2 instance. The frontend was written using HTML and CSS. The MySQL database holds the data for every post starting with the automatically assigned id field which auto-increments, then the name of the poster, then the post title, then the post body, then the post date, and lastly the name of the post image.

```
1  CREATE TABLE posts(
2          id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
3          name VARCHAR(45),
4          postimage VARCHAR(100),
5          title VARCHAR(45),
6          description TEXT,
7          postdate VARCHAR(100)
8  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figure 4: SQL definition for table holding post data

The role of NodeJS was to design all of the core network functionalities of the application. This specifically meant handling post and get requests from the client page, connecting to and communicating with the MySQL database, and setting up a port to listen for connections on. When the client first asks to connect to the server, the NodeJS handles a get request in which it uses the MySQL database to query for all entries in order of most recent to least.

```
43  app.get('', (req, res) => {
44
45
46      pool.getConnection((err, connection)=> {
47          if(err) return res.status(400).send({ success: false, err }); //not connected
48          console.log('Connected!');
49
50          //ORDER BY postdate DESC
51          connection.query('SELECT * FROM posts ORDER BY id DESC', (err, rows) => {
52              //Once done, release connection
53              connection.release();
54
55                  if(!err) {
56                      res.render('index', { rows });
57                  }
58                  else {
59                      return res.status(400).send({ success: false, err });
60                  }
61          });
```

Figure 5: Connecting to database using connection pooling then retrieving the query for all rows in specified order then sending it to Express-Handlebars before rendering it

This function then sends these rows of the database to an Express-Handlebars file which loops through each entry to display each post in html. The page also contains CSS which uses a grid format to create a page that is mobile friendly, and to make the page simple, appealing, and overall user friendly. The final step of the request is to render the complete html page and send it to the client. The functionality of the post request is that the client can send post information using an html form back to the NodeJS application.

```
<h1>Welcome to the Kettering Social Media Page!</h1>
<h2>Create Post Here!</h2>
<form action="/" method="POST" enctype="multipart/form-data">
    <h3>Post Title:</h3>
    <input type="text" id="title" name="title" maxlength="45"><br>
    <h3>Name:</h3>
    <input type="text" id="name" name="name" maxlength="45"><br>
    <h3>Post Body:</h3>
    <textarea id="pbody" name="pbody" rows="4" cols="50" style="white-space: pre-line; white-space: pre-wrap;"></textarea>
    <h3> Upload Image</h3>
    <input type="file" name="sampleFile" accept="image/*"/><br>
    <input type="submit" class="btn btn-primary">
</form>

<h2>Current Posts!</h2>
    {{#each rows}}
<div class="card">

    <h2 class="card__name">{{this.name}}</h2>
    <div class="card__title">{{this.title}}</div>

        {{#if this.postimage}}
            <img class="card__image" src='{{this.postimage}}' alt="Default">
            {{else}}
                <img class="card__image" src="/img/default.jpg" alt="Default">
        {{/if}}



    <p class="card__postbody">{{this.description}}</p>
    <p class="card__postdate">Uploaded: {{this.postdate}}</p>
</div><br><br>
    {{/each}}
```

Figure 6: Creating HTML form that will send input data to the server using a post request with code receiving the get request to display each post in the database below the form

The NodeJS receives an image file along with the string values of each field from the user's submitted post information. The image file is saved into a folder on the server after being given a unique name in that folder using Express-FileUpload. The name of that file along with the other data from the post request is then interested into the MySQL database along with a date.

```
106         //Use mv() to place file on the server
107         sampleFile.mv(uploadPath, function(err) {
108             if(err) return res.status(500).send(err);
109             });
110     }
111
112
113     pool.getConnection((err, connection)=> {
114         if(err) return res.status(400).send({ success: false, err }); //not connected
115         console.log('Connected!');
116             var dateObj = new Date();
117             var month = dateObj.getUTCMonth() + 1; //months from 1-12
118             var day = dateObj.getUTCDate();
119             var year = dateObj.getUTCFullYear();
120
121             const newdate = year + "/" + month + "/" + day;
122
123         var sql = 'INSERT INTO posts VALUES (NULL,"' + (req.body.name||'') + '","'
124         + (sampleFileDBname||'') + '","'+ (req.body.title||'') + '","' + (req.body.pbody||'') + '","' + (newdate||'')
125         +'")';
126
127
128         connection.query(sql, (err, rows) => {
129             //Once done, release connection
130             connection.release();
131
132                 if(!err) {
133                     res.redirect('/');
134                 } else {
135                     return res.status(400).send({ success: false, err });
136                 }
137         });
```

Figure 7: Sending the post data into the MySQL database and the image into a folder on the server

The page is then redirected to itself with the new post at the top of the page. This overall represents the system under which the web-application operates. This is a simple but effective way to make sure that the most recent post is then shown to the users screen, without the need for a refresh from the user.

IV. RESEARCH

A. *Node.JS*

Node.js is a javascript event driven runtime engine that drives network applications. Node.js is meant to be scalable as it can host many network connections. This is also a more efficient method than some others as it does not require threads which is simply more efficient and easier to use. It is also less likely to be blocked due to I/O unless there are functions also being performed synchronously within the Node.js library. Node.js also keeps HTTP in the forefront and is optimized for streaming and low latency. Another key reason we decided to use Node.js is that it integrates easily with our other components

B. *Express*

Express is a minimalist framework that is designed specifically for web applications. It comes with a plethora of API calls that interact with HTTP and the middleware. If the available API's do not meet your requirements, it is easy to create an API that will accomplish what is needed.

C. *EC2*

Amazon EC2 instances are virtual machines that are available to use on demand as needed. They are cloud services offered by Amazon that are generally Linux based, and can be used in all kinds of applications. Ours was being used as the server for our site, and was hosting the webpage, and MySQL database.

D. *Heroku*

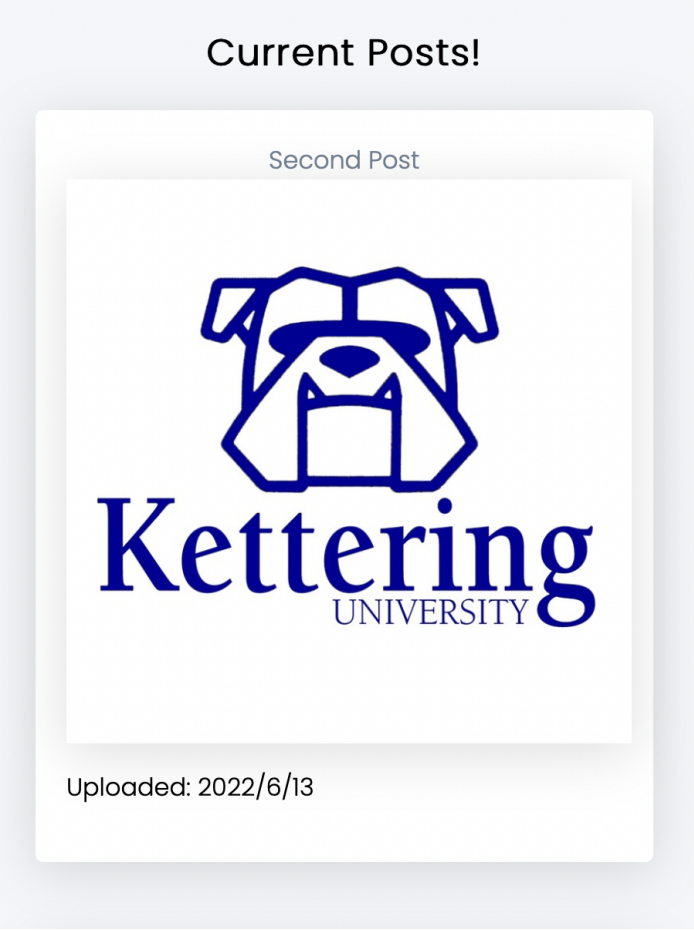
Heroku is a third party service that integrates with MySQL and Node.js and allows for easy communication between the two with support for multiple languages and easy integration. Heroku also offers a lot of manageability options that make it easy to work with our setup. They also offer a monitoring service on their products which allows for us to know if they are being used optimally and make any efficiency changes when needed.

E. *MySQL*

MySQL is an open source, relational database that has very wide usage options. It relies very heavily on the way the data is structured and the table structures within the schema. MySQL is one of the most popular databases out there and is used widely throughout the industry. It has a very simple syntax that is easy to pickup on even if unfamiliar with databases.

V. RESULTS

We ended up with a site hosted on an Amazon EC2 instance that has a form that takes the user input, and displays the most recent posts below. From the form, the user inputs 3 fields, a name, a title for the post, and a post description. Once they enter those fields, there is a select file button that allows for the user to select which photo they want to upload. Once the photo is chosen they submit their entries, and the submission is sent to the database. The submission is then displayed on the page at the top as the most recent submission. Our project is a success in our minds as we accomplished what we originally set out to do, which was providing a space where people have the opportunity to upload photos of things happening around campus or around them.



VI. LATER ADDITIONS

Some things that we could add in the future would be something along the lines of account creation, or having a way to track users and associate posts with people. This would allow for moderation of any bad users or anything along those lines. The account creation could hopefully be integrated into the current authentication system that Kettering uses as this is a Kettering focused application and would ensure all Kettering students have access to the site. Some other additions, a nice URL for the site would be very useful as the link currently is not very user friendly and a security issue as it is the IP and port being used by our EC2 instance. A cleaner URL would be easier for the user to remember, and would no longer be exposing the IP of our virtual instance. Some other smaller things for some extra functionality would be possible such as account searching to view posts, or possibly the addition of searching through the post body, title, or name. Other things, like getting the file format .heic working, which was giving us some troubles during testing and in the final product. Or possibly using a different storage system for the images, such as Amazon's S3 service as it would integrate well with our EC2 instance. Once we have integrated the accounts we could have a system set up to monitor the accounts to ensure that nothing bad or "illegal" is posted onto the website that could get users suspended for posting those things. Since the accounts will be linked to the kettering authentication system we could possibly see if what they post is bad enough for counselors/staff to be involved like if any student posted about potential harm to themselves or other students. We could add such things as group creation for a collection of users to post from a group such as the Frat houses using a group to talk about events and recruitment of other students. Group creation would also be nice for setting up a study group or just to have a group of friends in one location to discuss what you would like to do during your free time. I think that we should have maybe a group of staff for kettering to introduce the campus and events around campus for students to participate in on a weekly basis like how we have the events emailed to us through our kettering email it could be a read only post that could be saved to another part of the website labeled "Events" where fraternities and kettering staff could put events for students to participate in. Having the groups of "Events" and normal posts be separate would help the website not be so cluttered with random posts and have a centralized place for students to learn about events to be a part of.

VII. CONCLUSIONS

In conclusion we did what we set out to do which was creating a web application that would bring the community of Kettering together through the use of modern cloud computing technologies which allows for it to be easy and affordable for the average person to accomplish. As this website was created during the term, it shows just how much can be done with the technology that is available to us. Bringing people together through sharing images is a common concept that is now easily accessible to the entirety of the Kettering campus, centralized together. As this application is on the web and mobile friendly it allows for access to pretty much everyone as it is simply online and most people have a smartphone or some access to the internet. The website can also help students connect with other students like how during COVID most people ended up turning to social media such as “Discord” to connect with people around the campus. Even if we don’t have a situation like that again it is still nice to have an alternative way to communicate especially if you aren’t very good at doing so.

VIII. REFERENCES

[1] https://nodejs.org/en/about/

[2] https://expressjs.com

[3] https://www.heroku.com

[4] https://www.bezkoder.com/deploy-node-js-app-heroku-cleardb-mysql/