Stage IV - Design
Due: 3/30/2020
Jon Bernstein, Matt Perino, Numaan Cheema, Sam Bishop
GitHub repo: https://github.com/315-6/socialSrHub
Integration Lead: Numaan Cheema

- **Problem Statement**

The main issue we aim to solve involves the growth of the srhub.org and civicstory.org websites. The main issue we've been able to identify in the srhub.org and civicstory.org websites is the lack of community building. For a site so focused on events happening within communities for a good cause, the site lacks the ability to have people connect with one another and share content on topics that they hold an interest in. Another issue we've identified is a lack of reasons for people to keep returning to these websites.

- **Objective of the module.**

We aim to tackle the issue of community building by adding additional functionality that we believe will enhance the user experience, overall. By creating user pages, and allowing users to interact with other user pages (by allowing a user to set their pages to public, giving anybody the ability to see them or private, allowing only permitted individuals to see their page), we believe that we can allow users to build communities within this domain.

We believe our proposed module will also address the issue of reasons to return to the website by allowing users to see content that may be outside of their immediate realm of interest. By exposing users to new and interesting content, we believe that all users will have a reason to return to the sites often.

- **Description of the desired end product, and the part you will develop for this class.**

Our vision for the module we aim to develop is similar to that of a user's page on common social media sites, such as Facebook or Twitter. The end product will function similarly to these sites, allowing users to see content that users they "follow" or prominent figures within their communities are reposting or commenting on. We also would aim to recommend content to users, based on previous things they've looked at, potentially furthering their scope of interaction with the site.

As far as the module we will develop for class goes, our main priorities center around the setup for the user page, allowing users to favorite content they find interesting or useful, and potentially recommending additional content to users. Stretch goals include allowing users to see other user pages, and commenting and reposting posts.

- **Description of the importance and need for the module, and how it addresses the problem.**

We believe that development of the proposed module will ultimately increase traffic to srhub.org and civicstory.org, allowing for rapid growth of these websites. The ability for people to share things that they find important has proven useful for the growth of social media sites over the past few years and we think a similar principal can be applied here.

- **Plan for how you will research the problem domain and obtain the data needed.**

We've already performed some research into the problem domain by browsing other sites with similar goals and looking for features we found particularly useful or features that improved the user experience. In order to obtain the data needed, we will very likely use artificial data modeled after the content available on srhub.org and civicstory.org

- **Other similar systems / approaches that exist, and how your module is different or will add value to the existing system.**

Obviously, popular social media sites such as Facebook, Twitter, and LinkedIn were our main inspiration for development of our proposed module. Our module is unique in that it focused on the niche srhub.org and civicstory.org, which aims to provide users with resources and information relating to sustainability in New Jersey. We believe our module will add value to the site by allowing users to form connections through the context of these websites.

- **Possible other applications of the system (how it could be modified and reused.)**

Our system will provide a template for other small sites to implement similar systems. As previously mentioned, the main goal of our system is to allow users to build communities within the site and we think this can be applied to other sites with similar goals.

- **Performance -- specify how and to what extent you will address this.**

We will aim to maximize performance by performing as few queries as we can while accomplishing the functionality proposed. We believe this will be the main source of performance enhancement within the system. We are also researching alternative ways to maximize performance within this context.

- **Security -- specify how and to what extent you will provide security features.**

While we believe that the nature of the proposed system is relatively secure, we will provide some features that allow for additional security for the user and the system. For users, we will allow them to specify to what extent they want their page to be accessible by other users. For users who want only specified (or no one) to be able to view the content on their page, we will provide an option to make their page private. This function will allow the user to moderate who is able to see the information on their page.

In terms of system security, we will be parsing all user input to ensure that none of the common mistakes that applications like this have will appear in this system. One of the most common problems with systems that utilize a database is SQL injections, and parsing input along with a whitelist of accepted input will largely eliminate this threat.

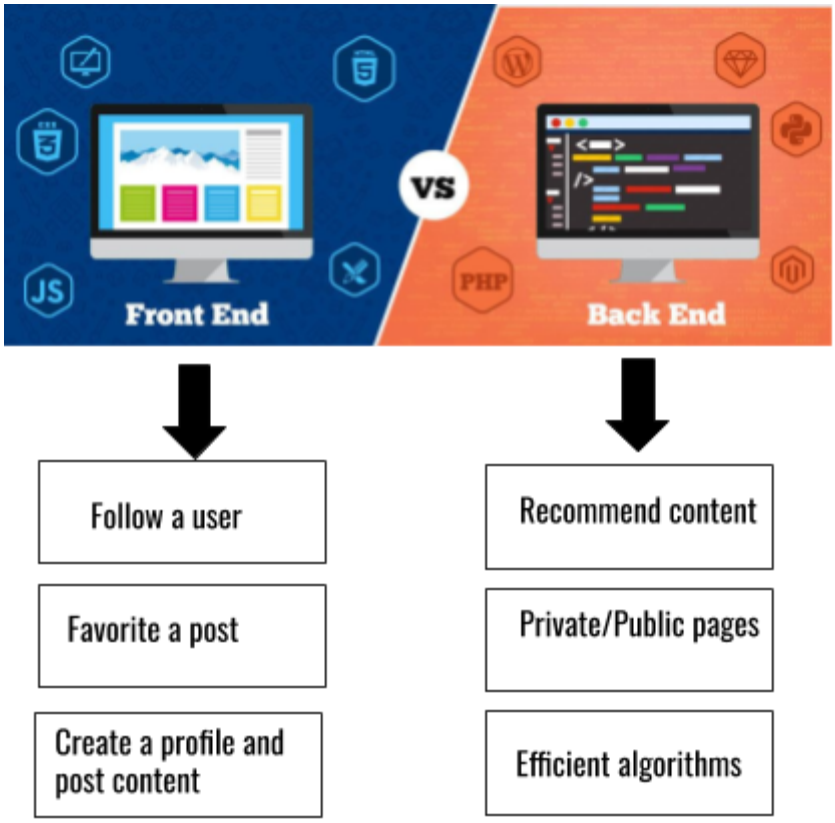
- **Backup and recovery -- specify how and to what extent you will implement this.**

In determining the most efficient way to ensure our system will remain backed up and recoverable is establishing what we need to be backing up. Each user will have a user profile that contains their personal information. This will be our main focus, preserving user data with the database. We will utilize logical backups, a system used in our DBMS. The server will save each user profile in a file within our VM, ensuring that if the original database fails, we have user information protected. We can automate this process within SQL, making backup and recovery a part of our system from very early on. Since we do not anticipate having tons of users data, along with other data, we will be able to store that data within the VM. If the amount of data we use exceeds that, we can switch over to hard disk data that is automatically updated over a set period of time.

- **Technologies and database concepts the team will need to learn, and a plan for learning these.**

The technology and concepts we will need is going to be the hardest task to accomplish. We are going to need to understand back end database development and how to efficiently store this data. This will be done through class along with supplementary work such as a deep dive into SQL and PHP. On the front end, we are going to need to familiarize ourselves with CSS and HTML to keep the user experience clean and easy to understand. In order for users to receive data similar to their favorites and other users they follow, some information retrieval and algorithms work is going to be used as well. The overall technology we are going to have to understand is how social networks work, front end and back end.

- **A diagrammatic representation of the system boundary that specifies what data you will model and which queries you will implement.**



TCNJ THE COLLEGE OF NEW JERSEY

Social Sustainability

Jon Bernstein, Matt Perino, Numaan Cheema, Sam Bishop

Objective

Build a community that allows users to connect with each other within the sustainability sector in New Jersey

Provide srhub.org and civicstory.org a way to connect with their users in a more technologically advanced way

Develop an easier way for srhub.org and civicstory.org to identify the leading problems in communities and the general populations opinions on sustainability in New Jersey

Building a Community

Approach

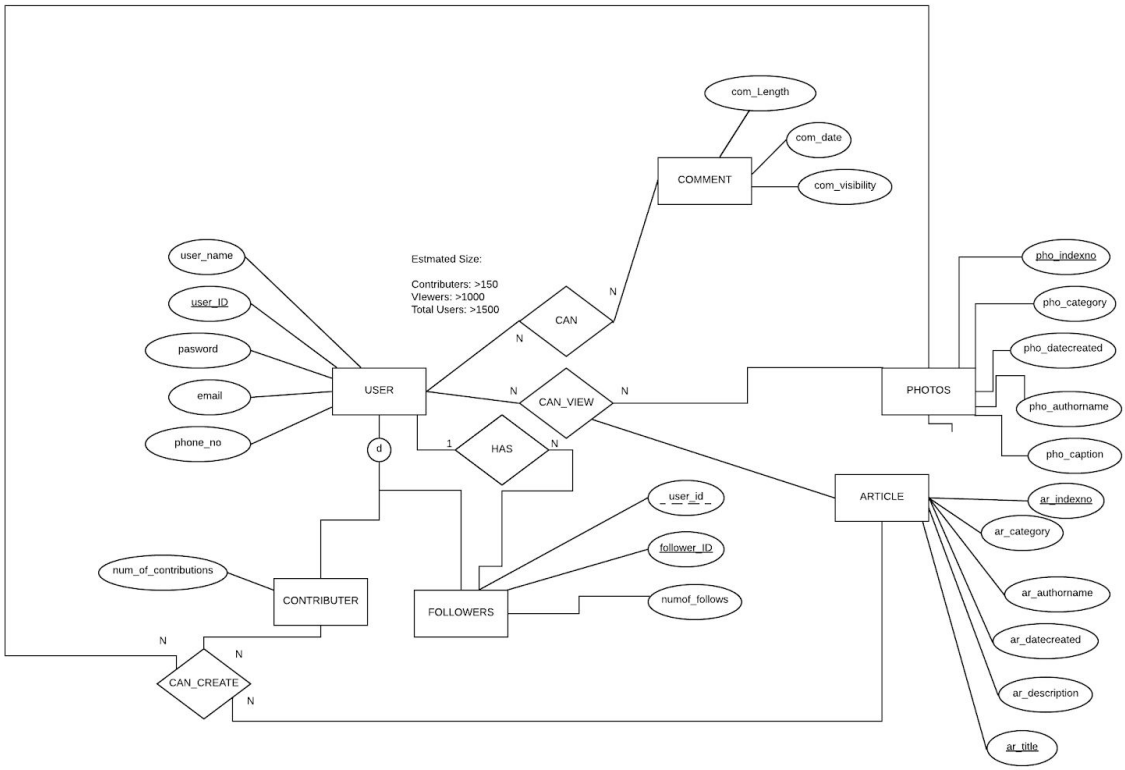
Create a community around shrub.org and civicstory.org

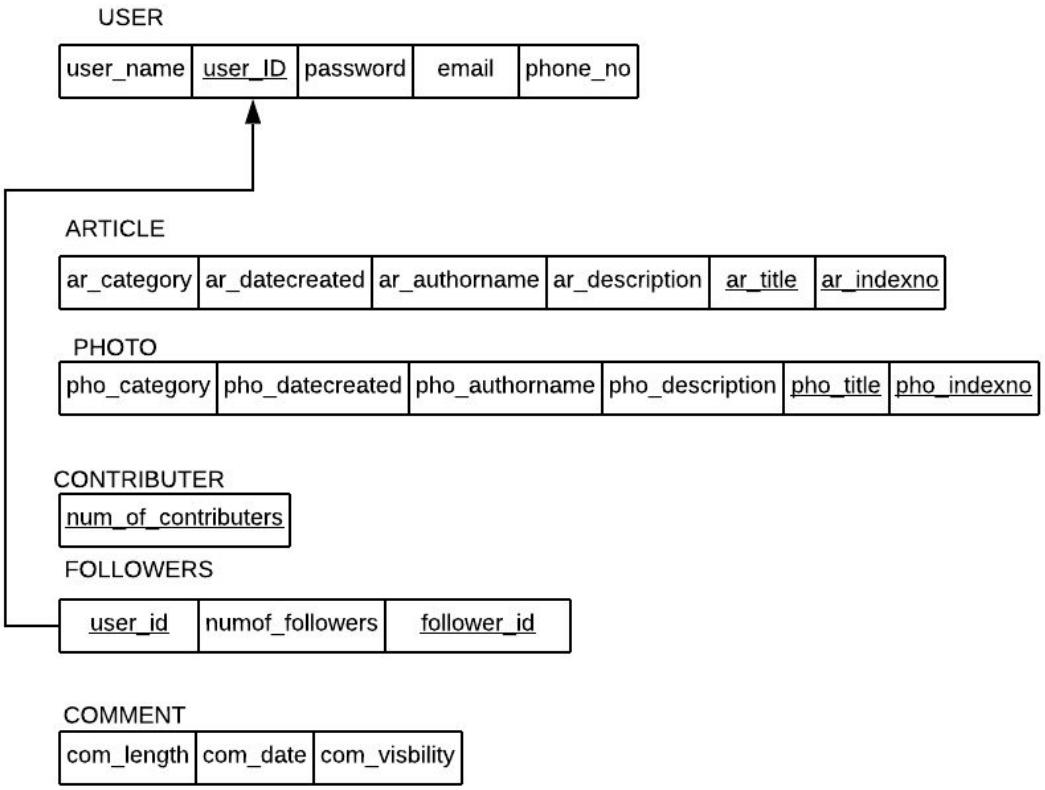
Key Milestones

• Project Specifications	02/24/20

Stage III - Database Model

Jon Bernstein, Sam Bishop, Matt Perino, Numaan Cheema





Stage IV
3/30/2020
Jon Bernstein, Numaan Cheema, Sam Bishop, Matt Perino

Demonstrate that all relations in the relational schema are normalized to Boyce-Codd normal form (BCNF).

- For each table, specify whether it is in BCNF or not, and explain why.
- For each table that is not in BCNF, show the complete process that normalizes it to BCNF.

USER

user_name	user_ID	password	Email	Phone_no

Satisfies the first three normalizations because:

1. All values are atomic, columns are unique, all data in each column is of the same type
2. There are no partial dependencies
3. There are no transitive decencies

It does also satisfy BCNF because:
user_ID → user_name
user_name → {password, email, phone_no}

ARTICLE

ar_category	ar_datecreated	ar_authorname	ar_description	ar_title	ar_indexno

Satisfies the first three normalizations because:

1. All values are atomic, columns are unique, all data in each column is of the same type
2. There are no partial dependencies
3. There are no transitive decencies

It does also satisfy BCNF because:
ar_title → {ar_datecreated, ar_authorname}
ar_indexno → {ar_category, ar_description}

PHOTO

pho_category	pho_datecreated	pho_authorname	pho_description	pho_title	pho_indexno

Satisfies the first three normalizations because:

1. All values are atomic, columns are unique, all data in each column is of the same type
2. There are no partial dependencies
3. There are no transitive decencies

It does also satisfy BCNF because:
pho_title → {pho_datecreated, pho_authorname}
pho_indexno → {pho_category, pho_description}

CONTRIBUTOR

num_of_contributor

Satisfies BCNF because:

1. All values are atomic, columns are unique, all data in each column is of the same type
2. There are no partial dependencies
3. There are no transitive decencies
4. For every functional dependency is dependant on a super key

FOLLOWERS

user_id	numof_followers	follower_id

Satisfies the first three normalizations because:

1. All values are atomic, columns are unique, all data in each column is of the same type
2. There are no partial dependencies
3. There are no transitive decencies

It does also satisfy BCNF because:
user_id → follower_id
follower_id → numof_followers

COMMENT

com_length	com_date	com_visibility	com_index

Satisfies the first three normalizations because:

4. All values are atomic, columns are unique, all data in each column is of the same type
5. There are no partial dependencies
6. There are no transitive decencies

It does also satisfy BCNF because:
com_index → {com_length, com_date, com_visibility}

Define the different views required. For each view, list the data and transaction requirements. Give a few examples of queries, in English, to illustrate.

For our project, we'll only need a few views, with more being required as we move towards our stretch goals for the semester.

For our user page, we'll need a view for the articles a user has liked as well as posts they choose to share on their page.

For the liked articles view, we'll need to ensure that the database keeps track of which articles are "liked" by a user, so that we do not allow the user to like the same article twice. We also need to do this to allow a user to revoke their like for a particular article. Additionally, we'll need to ensure that these views are user-specific. For example, we don't want any user to be able to influence the liked articles of another user.

The view for shared posts is a little more complicated. We'll need to keep track of the articles themselves, as well as other users' comments on these posts. We'll need to parse input to some extent to ensure there's no abuse of the input features provided to users. We will also consider putting restraints on comments, such as character limits and limiting how many comments a user can post on 1 article.

An example of the logic for the query that handles users liking a post follows: First, we need to make sure that the user has not liked this article previously. We do this by matching the article name, or some primary key for that article. If the article has already been liked, we enter a new flow that will remove that article from the liked articles. If the article has not been liked before, we will then enter the element into the users liked articles.

Design a complete set of queries to satisfy the transaction requirements identified in the previous stages.

For liking an article:
```
SELECT ar_title
FROM ARTICLE
where ar_title == selection
INSERT INTO LIKED_ARTICLES
VALUES(selection)
```

For commenting:
```
WHEN
        comment_text != blacklisted_words THEN INSERT INTO COMMENT(comment_text)
                                                VALUES(textbox)
        comment_text == blacklisted_word THEN exit
END;
```

To follow a user:
```
SELECT user_id
FROM USER
Where user_id == selection
INSERT INTO FOLLOWERS(follower_id)
VALUES(selection)
```

To like a photo:
```
SELECT photo_indexno
FROM PHOTO
Where photo_indexno == selection
INSERT INTO LIKED_PHOTOS(photo_indexno)
VALUES(selection)
```

To create an article:
```
CREATE article_no
FROM ARTICLE
Where article_title == NEW
INSERT INTO ARTICLES
```

To post a photo:
```
SELECT user_id
FROM USER
INSERT INTO PHOTOS(pho_authorname)
VALUES(user_id)
```