Group 9
Collaborative Project: Alternative to Civic Story

By:
CSC 315: Amulya Badineni, Adam Mellan, Julian De La Cruz, Alhassane Traore
JPW 321: Kieran Nashad, Joshua Camacho

NAME	TCNJ EMAIL ID	GITHUB ID
Amulya Badineni	badinea1@tcnj.edu	badinea1
Julian De La Cruz	delacrj4@tcnj.edu	delacrj4
Adam Mellan	mellana1@tcnj.edu	mellana1
Alhassane Traore	traorea2@tcnj.edu	traorea2
Kieran Nashad	nashadk1@tcnj.edu	YvngKiwi
Josh Camacho	camachj5@tcnj.edu	Jshua02

TABLE OF CONTENTS

Inception: “Executive Summary”

The CivicStory website is a great resource to find and receive news/information related to sustainability and creative change within the state of New Jersey. The website has a great deal of videos, articles, and podcasts that range from a whole host of topics such as art, environment, education, and many more. Although having all this information makes the website incredibly useful to New Jersey residents, it can be a bit difficult to find information on specific topics that a user might be interested in. It is for this reason that the group proposed a website that allows users to have easy access to the news of their choice.

In order to accomplish this goal, the group set out to create a website that would allow users to have a personal newsfeed populated with videos, articles, forums, and podcasts that pertain to the topics of their interest. To do this, the group first created a database consisting of four key relations named Account, Residence, Preferences, and Newsfeed. The account relation contains a user’s email address, password, first name, and last name. The residence relation consists of a user’s county, city, and zip code. The preferences table consists of two options the user can personalize: notification preference and region. The account, preferences, and residence relations have an auto-incrementing user identification number that keeps tracks of registered users. Lastly, the news feed relation consists of a media type, a link, main topic, and an overview of content. The news feed relation has an auto-incrementing media identification number to keep track of the media content in the database. As part of our user-interface, the group created a simple, intuitive website that aligns with their database. The website has four main pages: login, registration, preferences, and news feed. The login page allows previous users to sign in and update preferences. The preferences page allows users to update their account information, turn email notifications on/off, pick a preferred region, and delete their account. The registration page is mainly for new users who would like to become a part of the Civic Story family. And finally, the news feed page contains a table of all the media content in the database and has the ability to filter media content by tags, displaying only the ones designated by the user.

When comparing the original Civic Story website and the one created by this group, there are several key differences. To begin with, the homepage has less information, but still aims to represent the intent behind Civic Story through several icons (democracy, journalism, ecology, etc.). There is an account system for users to personalize their news feed. While the group’s website is certainly not a whole new website, it can be considered an extension following the concept of “less is more". The cost of creating the website is relatively nonexistent as it was created for educational purposes. Yet, both the webpages and the database can be scaled up through adding/modifying relations, attributes, and by adding features (filters, images, more media content) to the website.

Elaboration: Project Proposal & Specifications

I. Problem Statement
   A. The very first impression of CivicStory is overwhelming, its content seems to be thrown all around the website with no consideration to user demographics and preferences. The audience of the website is undefined. In other words, the content isn't personal to a user by location, interest, hobbies, or any other category. Users do not have a clear cut way to join the movement, other than viewing videos, forums, and articles.

II. Objective
   A. The goal of the project is to allow users to create an account and select specific preferences. By having accounts, the target audience of the website is defined. The content of CivicStory will be then filtered according to a user's preferences so that they can then get a custom feed that displays media content based on a certain tag. This feature can be enhanced by adding different types of filters.

III. Desired End Product
   A. Create a webpage that uses a database that contains user data and data from the CivicStory site to populate a custom news feed for each user.

IV. Description of Importance
   A. This will help CivicStory understand the type of content users enjoy and promote more relevant and accurate information in a timely manner. If users go to the site and find it difficult to find the content that they would most like to see, they most likely would not come back. Letting the users have custom feeds and allowing them to pick and choose what they do and do not like will enhance the overall user experience.

V. Plan to Research Information & Obtain Needed Data
   A. The students will read from the textbook as well as use the resources provided by the school and the internet to learn and understand the concepts needed to make this project successful. We will also search for other similar websites to better understand what the best method is for presenting newsworthy information to users.

VI. How it Can be Reused

A. A system just like this could be beneficial to many websites that have a large amount of content that users can access. For example, Netflix does something similar by asking new users what genres and movies/TV shows they enjoy and use that information to give them a unique and personal home page.

VII. Possible Other Applications

A. The intended end product could potentially be applied to a number of other applications. For example, if someone were to create a social media site from scratch, allowing for account creation and user data manipulation would be incredibly beneficial. This system could help in that aspect.

VIII. Performance

A. Accuracy - aim to present content that is relevant to the user's interests/metadata

B. Should be efficient enough to be scalable

C. Should be designed in a manner that makes future implementations as easy as possible.

IX. Security

A. According to the Github "Trust and Privacy" tab, they are "GDPR compliant" and obey the Privacy Shield Framework. The data we upload to Github will be encrypted and protected. As the project is created we will learn more about the extra security that we can implement.

B. Some security concerns that the team must keep in mind are proper data encryption/hiding so that the user can't see what they don't need to see, and making sure that user account information such as username, password, preferences, etc can only be accessed and modified by the user to whom that data belongs to.

C. The DBMS - we will decide what data can be accessed by which programs

X. Backup and Recovery

A. Logical backups will be performed periodically and saved to a different server in case the database needs to be rebuilt

B. Will have a dedicated backup server running in RAID 0

XI. Technologies & Concepts to Learn

A. Students will learn how to use DBSM (PostgreSQL) in order to store, retrieve and update data.

B. Tags - overlaps, constraints

C. Tables, Relationships, Cardinality

D. Data modeling

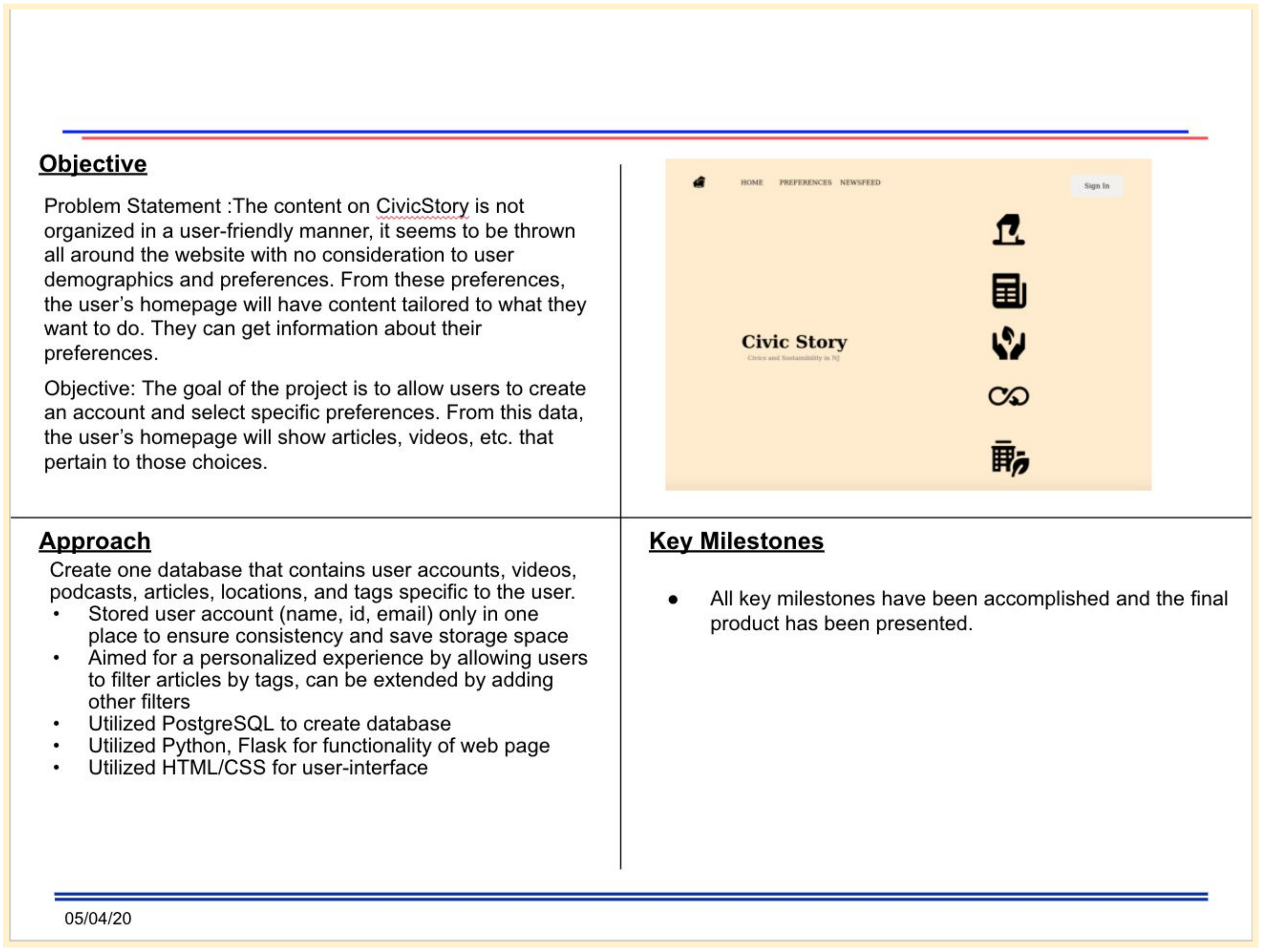
XII. Diagram

A. Please look at “Elaboration: Data Model”.

XIII. GitHub Project Link

A. https://github.com/CSC-315/cab-civic-story-group-9 ← Private

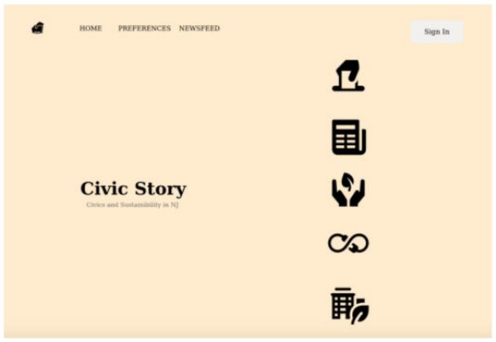
B. https://github.com/delacrj4/CSC315CivicProject ← Public



Objective

Problem Statement :The content on CivicStory is not organized in a user-friendly manner, it seems to be thrown all around the website with no consideration to user demographics and preferences. From these preferences, the user’s homepage will have content tailored to what they want to do. They can get information about their preferences.

Objective: The goal of the project is to allow users to create an account and select specific preferences. From this data, the user’s homepage will show articles, videos, etc. that pertain to those choices.



Approach

Create one database that contains user accounts, videos, podcasts, articles, locations, and tags specific to the user.

- Stored user account (name, id, email) only in one place to ensure consistency and save storage space
- Aimed for a personalized experience by allowing users to filter articles by tags, can be extended by adding other filters
- Utilized PostgreSQL to create database
- Utilized Python, Flask for functionality of web page
- Utilized HTML/CSS for user-interface

Key Milestones

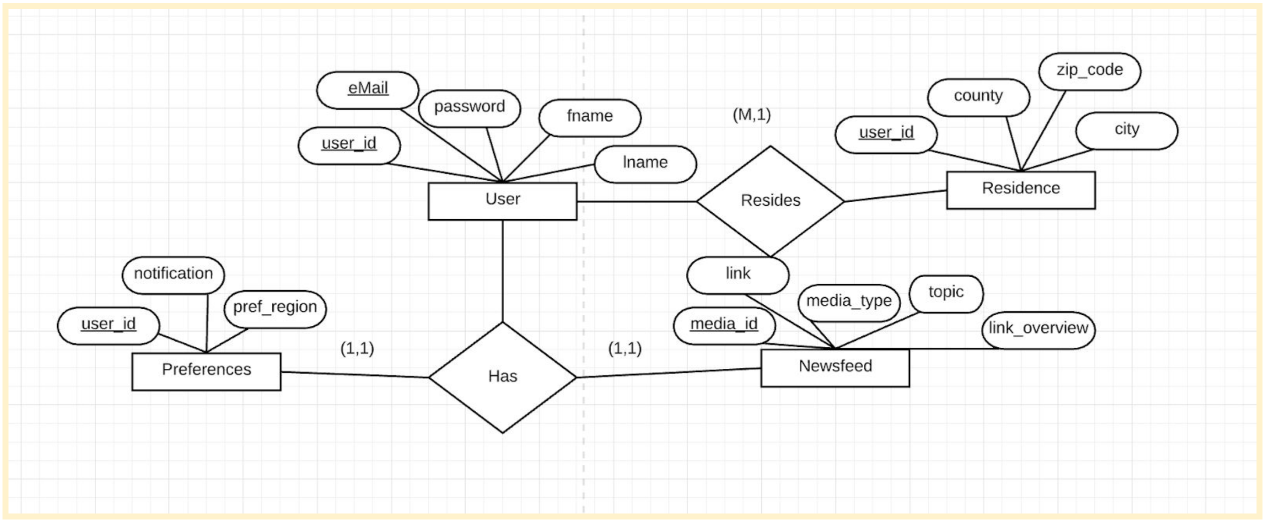
- All key milestones have been accomplished and the final product has been presented.

05/04/20

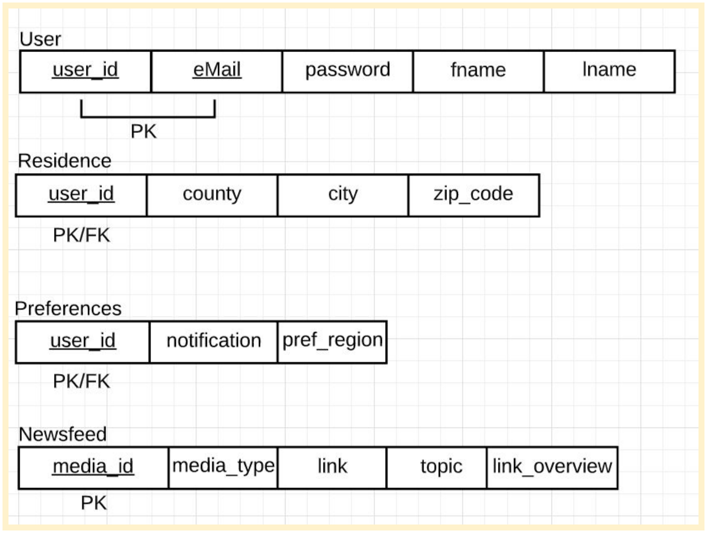
Elaboration: Design

PART ONE: Database Model

ER Diagram:



Relational Schema:



User

user_id	eMail	password	fname	lname

PK

Residence

user_id	county	city	zip_code

PK/FK

Preferences

user_id	notification	pref_region

PK/FK

Newsfeed

media_id	media_type	link	topic	link_overview

PK

Database Size:

- The initial database size was approximately 5 tuples for user accounts in order to perform tests. The database size increases/decreases based on the number of users registered on the website. The news feed table changes based on the insertions or deletions of a backend user.

- The number of records, in regards to users and pieces of media, will be small at the beginning but gradually increase as the popularity of the website grows. For the regions, it will be a fixed number that represents the residential information about New Jersey. However, the approach we used for one state can be applied to another one.

Types & Average Number of Searches:

- Users, for now, can filter articles by a tag(s). This feature can be enhanced by adding several other filters such as for lot (media type), keywords from link overview, article number, and by region.

PART TWO: Normalization

<u>Table 1</u>

1. ACCOUNT

<u>eMail</u>	password	fname	fname	county	city	zip_code

  - This table is not in BCNF because of data redundancy, a deletion anomaly, and its form is only in 1NF. The table does satisfy these 4 requirements of 1NF: each column has atomic values, the values under each attribute are of the same type, the column names are unique, and the order of values doesn't matter. The first issue to address is data redundancy. The attributes *county*, *city*, and *zip_code* can add repetition of similar data in multiple places and also cause anomalies. A deletion anomaly can occur if we have a user whose county is unique. For example, say the database contains one individual from Mercer County. If the user decides to delete his/her account, then that would cause a deletion in not only the user's account credentials but also his/her residential information that has not been saved in any other table. And so, we would accidentally lose a related dataset in the process of deleting another dataset. To fix the anomaly, we create another table where only the residential information is stored (as shown in the next bullet point).
  - RESIDENCE (the table as a result of the normalization of ACCOUNT):

<u>user_id</u>	county	city	zip_code

- ACCOUNT (the modified table after normalization of ACCOUNT):

user_id	eMail	password	fname	fname

2. ACCOUNT (normalized)

user_id	eMail	password	fname	lname

- The table is now in BCNF because there is no data redundancy, partial dependencies have been removed by creating another table, and transitive dependencies do not exist as the attributes depend on the composite key (user_id + eMail) can uniquely identify any row of the USER table.
- FUNCTIONAL DEPENDENCIES:
  - user_id eMail → password, create_date, fname, lname

3. RESIDENCE (after and from part 1) (normalized)

user_id	county	city	zip_code

- The table is in BCNF because there is no data redundancy, partial dependencies do not exist as all the attributes are dependent on the primary key, and transitive dependencies do not exist as the attributes depend on the primary key (user_id) can uniquely identify any row of the Residence table.
- FUNCTIONAL DEPENDENCIES:
  - user_id → county, city, zip_code

Table 2

1. PREFERENCES

user_id	notifications	pref_regions

- This table is in BCNF because all attributes depend on the primary key, which leaves no room for any non-prime key to depend on another non-prime key.
- FUNCTIONAL DEPENDENCIES:
  - user_id → notification, pref_region

Table 3

1. NEWSFEED

media_id	link	media_type	topic	link_overview

- This table is in BCNF because all attributes depend on the composite primary key of media_id and link. The attribute media_id would increment every time a new piece of media, an article, video, or image, is uploaded. So, each piece of media has a unique identification number and a link that will allow for an easy retrieval process.
- FUNCTIONAL DEPENDENCIES:
    - media_id, link → media_type, link, topic, link_overview

PART THREE: Views

1. To ensure the security of information from unauthorized users, we will be using query authorization statements, GRANT and REVOKE, in order to hide certain tuples. The information that will never be seen by a user is the password, user_id, and media_id. These are for the backend users only.
2. One view that will be needed is the join between User, Preferences, and Newsfeed. This output is important because it ties together a user's preferences with their sign in and newsfeed information.
    a. Data: user's password and ID as well as their preferences and newsfeed information
    b. Transaction: Verification of user's password and ID in order to access their preferences and display the correct newsfeed
    c. Query: for this transaction would be to select the user ID and password to verify they are the correct pair to login with.
3. Since the view is supposed to be always up-to-date, the preferred way of efficiently implementing a view would be through view materialization. The immediate update strategy, for updating any changes made in the base tables appear in the view, seems to be the best decision for our database. Mainly because we assume that most of the information provided by the user will remain constant with the exception of a few changes to per year. Since the changes are few and will most likely happen around the same time, using the immediate update strategy will be less complex, quick, and relatively easy.
4. To implement user interaction, a user can:

    a. search for articles specified by a tag
    b. delete his/her account
    c. choose a preferred region
    d. update account information
    e. turn on notifications
5. The database administrators are the only ones who can INSERT, UPDATE, and DELETE contents of the newsfeed relation from the backend.

Construction: Tables, Queries, and User Interface

CREATE DATABASE

```
psql CREATEDB civicStoryDB;
```
* civicStoryDB is the name of the database. You may change it to anything else

```
Sudo postgres -u psql
ALTER USER osc WITH SUPERUSER;
ALTER USER osc WITH CREATEROLE;
ALTER USER osc WITH CREATEDB;
ALTER USER osc PASSWORD 'osc';
```
* osc is the username. Yours may be different. Password is also osc which you can change to anything else

TABLES

```
DROP TABLE ACCOUNT CASCADE;
DROP TABLE RESIDENCE CASCADE;
DROP TABLE PREFERENCES CASCADE;
DROP TABLE NEWSFEED CASCADE;

CREATE TABLE ACCOUNT
(user_id SERIAL UNIQUE,
eMail VARCHAR (355) NOT NULL,
password VARCHAR (10) NOT NULL,
fname text,
lname text,
PRIMARY KEY(user_id, email));

CREATE TABLE RESIDENCE
(user_id SERIAL NOT NULL,
county text NOT NULL,
city text NOT NULL,
zip_code VARCHAR(5) NOT NULL,
FOREIGN KEY (user_id) REFERENCES ACCOUNT (user_id));
```

```
CREATE TABLE PREFERENCES
(user_id SERIAL NOT NULL,
notification text,
pref_region text,
FOREIGN KEY (user_id) REFERENCES ACCOUNT (user_id));
```

```
CREATE TABLE NEWSFEED
(media_id SERIAL NOT NULL,
media_type text NOT NULL,
link text,
topic text,
link_overview text,
PRIMARY KEY(media_id);
```

INSERTIONS

-
  ```
  INSERT INTO ACCOUNT (user_id, eMail, password, fname, lname)
  VALUES (DEFAULT, 'john123@company.com', 'Password1', 'John', 'Smith');
  ```

-
  ```
  INSERT INTO RESIDENCE (user_id, county, city, zip_code)
  VALUES(DEFAULT,'Mercer', 'Ewing', 08618);
  ```

-
  ```
  INSERT INTO PREFERENCES (user_id, notification, pref_region)
  VALUES (DEFAULT,'no', 'Middlesex');
  ```

-
  ```
  INSERT INTO NEWSFEED (media_type, link, topic, link_overview)
  VALUES ('Forum', 'https://www.civicstory.org/ecology-economy',
  'Economy', 'Five panelists explore such questions as: what is an ethical
  response to climate change, can news help us live better, and is growth
  always good?');
  ```

QUERIES:

-
  ```
  CREATE VIEW accountV as
  SELECT fname, lname
  FROM ACCOUNT;
  ```

-
  ```
  CREATE VIEW residenceV as
  SELECT  county, city, zip_code
  FROM RESIDENCE;
  ```

-
  ```
  CREATE VIEW preferencesV as
  SELECT notification, pref_region
  FROM PREFERENCES;
  ```

-
  ```
  CREATE VIEW newsfeedV as
  SELECT media_type
  ```

```
FROM NEWSFEED;
```

-
  ```
  SELECT fname
  FROM ACCOUNT
  WHERE lname = 'Smith';
  ```

-
  ```
  SELECT fname, county
  FROM ACCOUNT, RESIDENCE
  WHERE ACCOUNT.fname = 'John' AND RESIDENCE.county = 'Mercer';
  ```

-
  ```
  SELECT link
  FROM NEWSFEED
  WHERE media_type = 'video';
  ```

-
  ```
  SELECT county
  FROM RESIDENCE
  WHERE user_id IN
          (SELECT user_id
          FROM PREFERENCES
          WHERE RESIDENCE.county = PREFERENCES.pref_region and
          RESIDENCE.user_id = PREFERENCES.user_id);
  ```

-
  ```
  SELECT fname, lname, county, city
  FROM ACCOUNT
  INNER JOIN RESIDENCE ON RESIDENCE.user_id = ACCOUNT.user_id;
  ```

-
  ```
  SELECT fname, lname, email
  FROM ACCOUNT
  INNER JOIN RESIDENCE ON RESIDENCE.user_id = ACCOUNT.user_id
  WHERE county = 'Mercer';
  ```

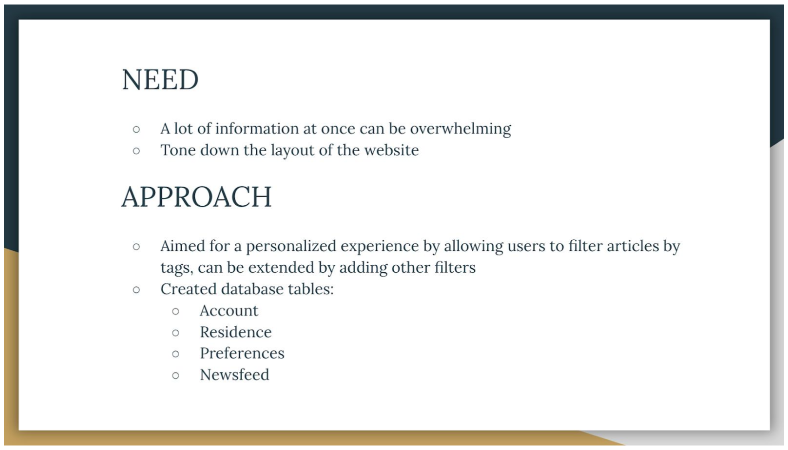
USER INTERFACE

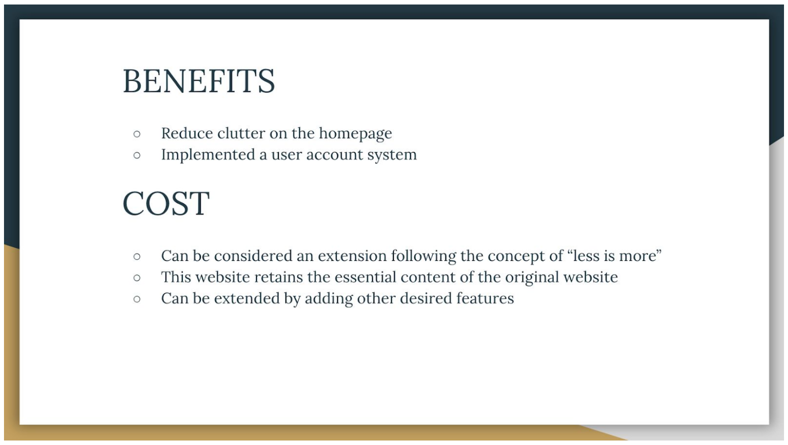
- Source code for user interface is on GitHub (look at “Transition: Maintenance”)

Transition: Maintenance

- GitHub Project Links:
  - Private → https://github.com/CSC-315/cab-civic-story-group-9
  - Public → https://github.com/delacrj4/CSC315CivicProject

Transition: Product Handover

- Presentation
  - 
    NEED
    - A lot of information at once can be overwhelming
    - Tone down the layout of the website

    APPROACH
    - Aimed for a personalized experience by allowing users to filter articles by tags, can be extended by adding other filters
    - Created database tables:
      - Account
      - Residence
      - Preferences
      - Newsfeed
  - 
    BENEFITS
    - Reduce clutter on the homepage
    - Implemented a user account system

    COST
    - Can be considered an extension following the concept of "less is more"
    - This website retains the essential content of the original website
    - Can be extended by adding other desired features
- Link to public GitHub project

  Public → https://github.com/delacrj4/CSC315CivicProject