Group 1

Team/Project Name: Trentoniana Catalog System

Team:
Thomas Orth: ortht2@tcnj.edu
Kathleen Burke: burkek11@tcnj.edu
Justin Pabon: pabonj3@tcnj.edu
Matthew Van Soelen: vansoem1@tcnj.edu

Inception: Summary

The need that our team identified is for a more user friendly system for the Trentoniana transcripts. The stakeholders that were identified for this project would be the group that currently maintains the Trentoniana transcripts collections and researchers/users who require access to the system in order to perform their daily/necessary tasks. There is a need for an easier process for maintaining and adding transcripts to the Trentoniana collection as well as a need to give users more in-depth information about the transcripts.

The approach our team took to satisfy this need was to create a web application, centered around the Trentoniana transcripts. Other web services such as the Internet Archives is built to be flexible to support many different types of data but this causes a loss in the user experience for someone looking for information on the Trentoniana transcripts. Our approach consists of a three tiered architecture approach. The first tier is the web client that a user would see. By offering an interface that follows the principles of good design and strives to create a good user experience, users can intuitively understand it and find the relevant transcripts they need. The second tier is our application server. Our application server allows for user authentication, handling of a regular user vs a librarian, searching of transcripts and its related information. The server robustly handles all queries and transactions it sends to the database. We also have laid the groundwork to make this system have customizable deployments in order to allow for the changing needs of the stakeholders to be met in terms of deployments. Our third and final tier is the database server. We used a modern database management system in order to safely store all information needed.

There are many benefits offered by our system. First, by designing it around the Trentoniana transcripts, we can offer a personalized user experience that makes the system easier to use. Secondly, our system offers more information for each transcript entry than the current system. We can provide information such as the participants, locations, and keywords that pertain to a transcript entry. Next, we also provide the ability for users to download the transcript files and bookmark them to reference later. We also provide the ability for librarians to upload new transcripts, update existing ones, or delete transcripts from the system. Based on our design, we provide all this in a streamlined application.

The cost to the stakeholders that maintain the Trentoniana transcripts is that this system is intended to replace the current website. This would require figuring out deployment of the new system and hosting. Additionally, as this project still requires work, there would need to be developers working on it after this submission to stakeholders in order to ensure that it is production ready.

Elaboration: Project Proposal and Specifications

Revisions are shown in red

Proposal

- **Problem statement**

  Within the Trentoniana Collection and the connected Internet Archive, there is a lack of an easy to use interface to favorite or bookmark transcripts. Additionally, there seems to be no way to bulk download transcripts that you like. This is significant because for research purposes, a user may want to download a large quantity of transcripts to examine. By only being allowed to download one at a time, this hinders the research process and could be sped up by bulk downloads. Each transcript entry seems to have a lack of information associated with it, which makes it hard for the user to tell if they found the right transcript.

- **Objective of the module**

  The objective of the module we will design is to allow for users to favorite transcripts that they search for. The module would also provide a cleaner interface to allow for easier use and clearer search results. It would provide a better user experience overall while searching for transcripts. The user experience would be improved by applying principles of good user interface design to ensure that the interface is intuitive and easy to use. The module would also provide more contextual information for each entry so that the meaning of that entry is clear to the user. It would help users be able to efficiently conduct research and access necessary files.

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

  The end product would be a web application. It would support 2 user types. First, a general user that can view transcripts, download them and favorite them to refer to later. The second user would be a librarian or other administrator who could update existing transcripts as needed and add new ones. The search functionality would be able to filter by keyword in title, keyword in the transcript, keyword in summary, and by participant names in the transcript entry.

  Each transcript would have an audio file, title, transcript text, transcript file, summary, a list of participants in the transcript, and the locations that are mentioned in it to ensure that each entry has descriptive information.

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

There is an importance and need for this module to ensure that there is an easy way to retrieve these transcripts. There is a rich history with the Trentonia transcripts and there needs to be a module that provides an informative way to understand these transcript entries. Our module addresses this problem by providing an easy to use interface as well as a more detailed entry description.

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

We plan to research the domain area in two ways. The first way is by exploring the website provided to us during the first collaborative meeting: trentonlib.org/trentoniana/audio-visual/. Secondly, we will explore the domain area by discussing what occurs in LNG-371/HON 270 since the course is directly related to the transcripts. We will obtain the data through the transcripts provided by Dr. Steele to the LNG/HON students so that we can store this into our database.

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

The current similar system that exists is the internet archive that is linked on the trentonian section of the trenton public library website. Our module will be different in the sense that it is currently only meant to hold data in the specified format that the transcripts have. The internet archive stores many different types of data and information that can bloat the search results and make it hard to find the right things. As a result of this, our module will be more user friendly and less complicated since we are only supporting one type of data. The American Folklife Center (https://www.loc.gov/folklife/states/northcarolina.html) links a bunch of separated databases but our solution would allow a cohesive search through all our documents. The schomburg center (https://www.nypl.org/locations/schomburg) is a catalog system that only tells you the presence of the mostly physical elements. For the digital components, it required the use of outdated technology such as the flash player. Our module would allow for direct reading and listening to transcripts (supported by our storage of the audio file and transcript text). We also would build our module with more modern technology in order to develop a modern system.

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

Since this system needs to support authentication, the user and librarian authentication services could be used to authenticate other applications for the organization. Additionally, if the participants of the transcript are important to other sources of data, the participants data, which will be stored in its own table, could be used by some other application if we exposed it through an API endpoint.

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

We will address performance by optimizing all our queries as we study more and more about databases. We anticipate that our queries to start off may be inefficient in some areas but once we progress into the semester and the project, we expect to be able to upgrade all the ones that can be made more efficient.

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

Given the nature of the data, the only private information that will need to be encrypted is the password for the users. This is due to the transcripts being publicly available. We will ensure proper rights and access to data by authenticating all our routes so that no one can access information or perform actions that they do not have the privilege to do.

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

We can write scripts that will support both backup and recovery. Postgres has support for data backup and recovery, as described here: https://www.postgresql.org/docs/9.1/backup.html. Additionally, the upload files can be zipped and stored separately and be unzipped and moved to the proper directory during recovery.

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

  There are three categories for technology that we will need to familiarize with:
  - Database
    - We will be using postgres: https://www.postgresql.org/docs/
  - Back-End
    - We will be using Python and Flask as an application server layer. We also would need a library to interact with the database. We will explore SQLAlchemy and psycopg2.
    - For Python: https://www.udacity.com/course/introduction-to-python--ud1110
    - For Flask: https://flask.palletsprojects.com/en/1.1.x/tutorial/
    - SQLAlchemy: https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/
    - Psycopg2: https://www.postgresqltutorial.com/postgresql-python/
  - Front-End
    - We will be using HTML, CSS and the built in templating engine for Flask. We will learn these through w3schools.com

  There is also a number of database concepts we will have to learn
  - Queries
  - Relationships

        - We expect to have 3 N:M relationships
            - N:M between a User and a Transcript
            - N:M between a Transcript and a Participant
            - N:M between a Transcript and Location
    - Optimizing Queries
    - Optimizing database structure for efficiency and scale

For these concepts, the 315 students will learn this through the class and ensure that all group members have at least a surface level understanding of the concept, depending on their role in the team.

- A diagrammatic representation of the system boundary that specifies what data you will model and which queries you will implement. The one provided is updated to include information about location information (next page)

Module as a whole

System Boundary

User (Model/Role)
Search, Favorite and view transcripts

Librarian (Model/Role)
Add, Update, and Delete Transcripts

Participant (Model)
Name of a person involved in a transcript

Location (Model)
Location text mentioned in a transcript

Transcript (Model)
Contains information regarding the transcript and its recording

Queries to be implemented:
Search by keyword (summary, main text, title)
Filter by participant(s)
Show a selected transcript, download text entry
User bookmarks transcript
Show a User's bookmarked transcripts
Create new transcripts, update transcripts, delete them
Add new participants, delete them
Add Location, delete them

Admin
Run backups from UI
Manage Users

Additional Transcript Data:
Timestamps of when participants speak. Add images for the locations mentioned in the transcript

Additional Functionality:
Bulk Download Transcripts
Annotate transcript.
Display location pictures

- 1-page quad chart (See next page)

Providing a User Friendly Module for Transcript Data

Thomas Orth, Matthew Van Solen, Justin Pabon, Kathleen Burke

Need

- Current website is hard to use
- Too much other data stored current solutions

Approach

- Design a simpler interface
- Allow for users to bookmark favorites
- Provide more information about each entry to allow for easier searches

Benefit

- Increased User Experience
- More robust handling of data
- Easily obtain the necessary trans -cript

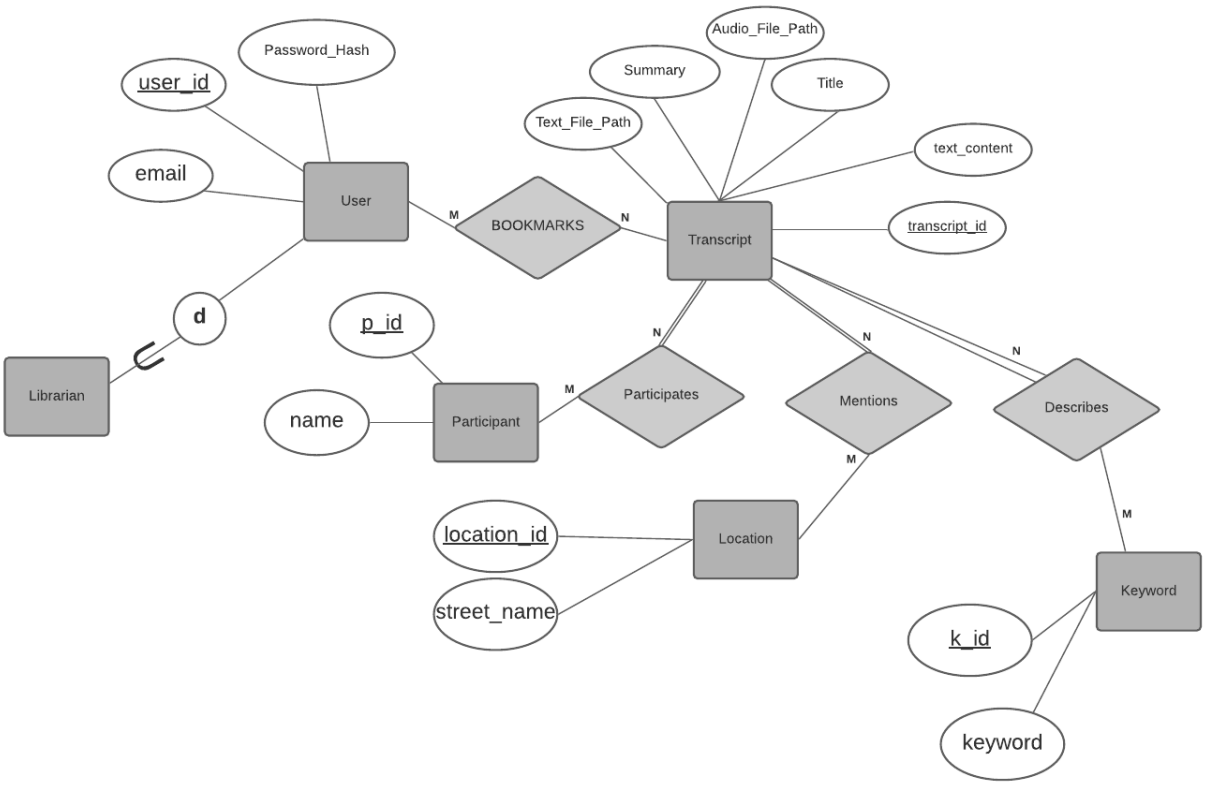
Competition

- The interface would be superior
- The entries would be more informative
- The search results would not encumbered by unrelated data

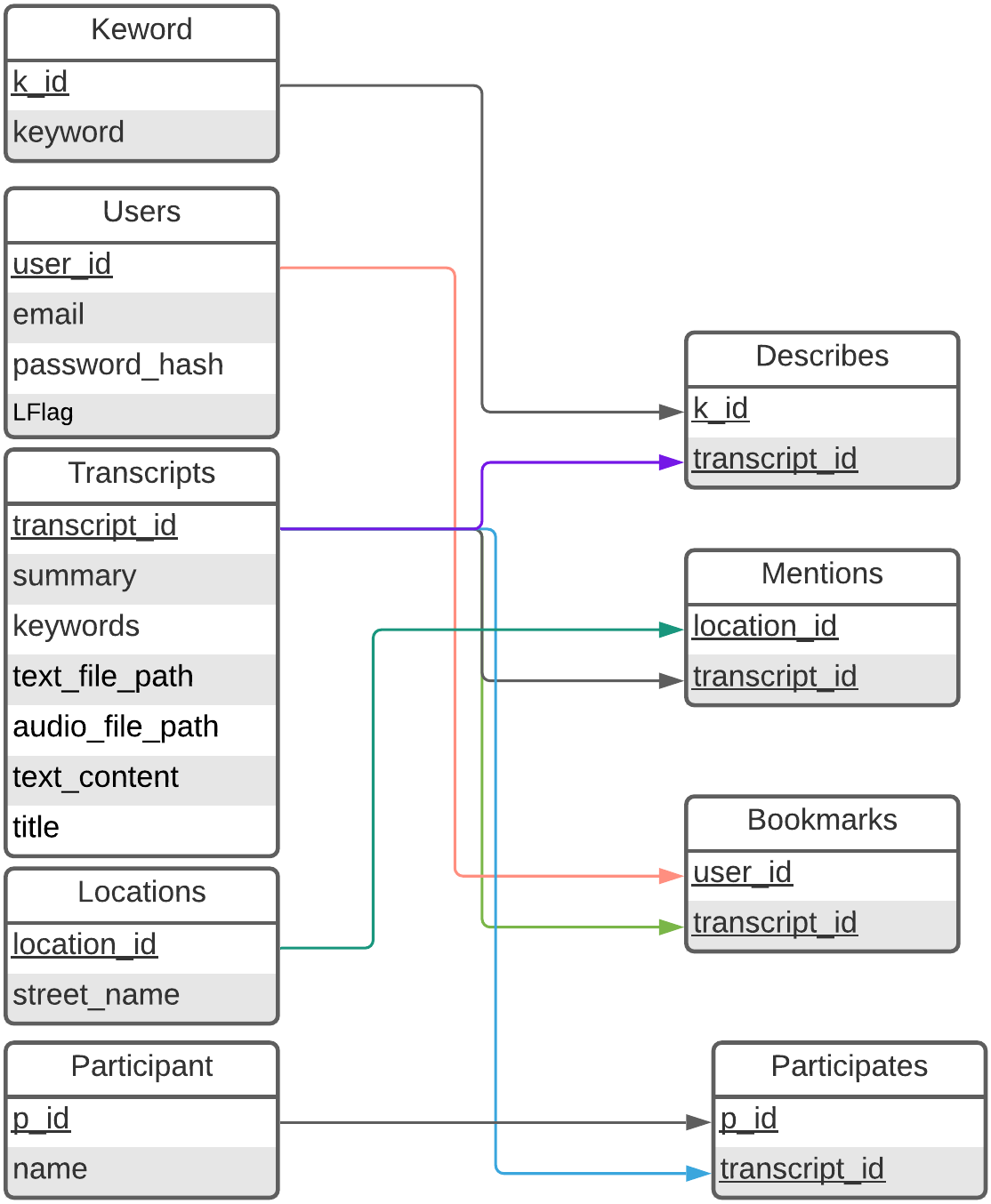
Elaboration: Design

Both diagrams below were updated to reflect a change to make keywords that describes a transcript act as a separate entity for the transcript.

EER Model



Relational Schema



Initial Database Size:

Assumption: We will be testing with a couple transcripts during the semester. Let's assume 5 of them are used. Lets also assume each transcript has 2 unique participants each and 3 unique

locations. Another assumption is we will only have 3 users to start: 2 regular users and a librarian. They will be added via our web client and server. Thus, the size of our initial database is:

2 users
5 transcripts
10 participants
15 locations
15 mentions entries (since each transcript has unique locations by our assumption)
10 participates entries (since each transcript has unique participants by our assumption)
0 bookmarks entries (Our initial database size will not have had a user bookmark anything yet)

Types of searches and numbers per each:

We anticipate the main searches being the following types:

- Filter by partial match in title
- Filter by participants
- Filter by partial match in text
- Filter by partial match in summary
- Filter by location
- Filter by keywords

Assuming that at a given time, there are about 25 researchers or users who need access to the transcripts. Let's assume that at this point, there are many transcripts entered into the system and the users are required to review many of them. Each of these are important searches and filters for a given user. Therefore, it is necessary for them to do these searches a couple times in a given day. If each researcher or user were to perform each search 5 times a day, to find the necessary transcripts, each search would have 125 executions each day.

Elaboration: Design

Table Examination for BCNF compliance:

Users Table:

This table is in BCNF due to the following. Take the following superkeys that are available in User:
Email
User_id
Password_hash
These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. That leaves the only non-prime attribute columns to LFlag. Note the following Dependencies:
Email → User_id
Email → Password_hash
Email → LFlag
User_id → Email
User_id → Password_hash
User_id → LFlag
Password_hash → Email
Password_hash → User_id
Password_hash → LFlag
There are no columns that are Functional Dependencies of LFlag. Therefore, this table is in BCNF.

Transcripts Table:

This table was originally not in BCNF as it had violated a 1NF constraint of no multivalued attributes. Upon review, we had determined that keywords were essentially a multivalued attribute. We split it from the transcripts table into a separate “Keywords” table and connect it to the Transcripts table via the “Describes” relationship.

This table is now in BCNF due to the following. Take the following superkeys that are available in User:
transcript_id (Primary Key)
text_file_path (candidate key)
audio_file_path (candidate key)
text (candidate key)
title (candidate key)
summary (candidate key)
These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. There are no non-prime attributes in this table. That is because each transcript is considered unique for these attributes and

therefore, there cannot be duplicate values in the tuples. This means that this table only consists of candidate keys/prime attributes. The table/relation is therefore in BCNF.

Locations Table:
This table is in BCNF due to the following. Take the following superkeys that are available in User:
location_id (Primary key)
street_name (candidate key)
These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:
location_id → street_name
street_name → location_id
The only FD that exists is between superkeys. Therefore, it is in BCNF.

Participants Table:
This table is in BCNF due to the following. Take the following superkeys that are available in Participants:
p_id (primary key)
name (candidate key)
These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:
p_id → name
name → p_id
The only FD that exists is between superkeys. Therefore, it is in BCNF.

Keywords Table:
This table is in BCNF due to the following. Take the following superkeys that are available in Keywords:
k_id (Primary key)
keyword (candidate key)
These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:
k_id → keyword
keyword → k_id
The only FD exists between superkeys. Therefore, this table is in BCNF.

Mentions Table:

This table is in BCNF. It consists solely of its primary key, (transcript_id, location_id), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Bookmarks Table:

This table is in BCNF. It consists solely of its primary key, (transcript_id, user_id), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Participates Table:

This table is in BCNF. It consists solely of its primary key, (transcript_id, p_id), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Describes Table:

This table is in BCNF. It consists solely of its primary key, (transcript_id, k_id), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Database Views:

We would need three views:
Participant_transcript_view
This is a joining of Participants and Transcripts tables based on the Participates relation.
Queries:
Joining with the Location_transcript_view and Keyword_transcript_view to have a full joined database (Full_Transcript)
User_transcript_view
This is a joining of transcript and user tables based on the bookmarks relation.
Queries:
Select all entries that match a given user_id
Filter on the above selected entries based on partial matching of title, summary or other transcript attributes
Location_transcript_view
This view is a joining of the Locations and Transcripts tables.
Queries:
Joining with the Participant_transcript_view and Keyword_transcript_view to have a full joined database (Full_Transcripts)
Keyword_transcript_view
This view is a joining of the Keywords and Transcripts tables
Queries:
Only the aforementioned joining
Full_Transcript_view
This view is a joining of Participant_transcript_view and Location_transcript_view
Queries:
Filtering on partial or full matches of title, summary, keywords, tex_content, name (participant), word (keyword) or street_name

Examples:
Find all transcript information whose title partially matches “trenton area”
Find all transcripts whose participant’s names partially match “joel”
Find all transcripts whose mentioned locations partially match “main street”
Find all transcripts who has the keyword “urban” that describes the transcript
Find all transcripts whose text content contains “welcome to today’s meeting”

Initial Queries (Revisions shown in red):

CREATE TABLE and CREATE VIEW queries

```
CREATE TABLE transcripts (
  transcript_id SERIAL PRIMARY KEY,
  title text UNIQUE,
  summary text UNIQUE,
  audio_file_path text UNIQUE,
  text_file_path text UNIQUE,
  text_content text UNIQUE
);

CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  email text UNIQUE,
  password_hash text UNIQUE,
  lflag int
);

CREATE TABLE participants (
  p_id SERIAL PRIMARY KEY,
  name text UNIQUE
);

CREATE TABLE locations (
  location_id SERIAL PRIMARY KEY,
  street_name text UNIQUE
);

CREATE TABLE keywords (
  k_id SERIAL PRIMARY KEY,
  keyword text UNIQUE
);

CREATE TABLE bookmarks (
  user_id int REFERENCES users,
```

```
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (user_id, transcript_id)
);


CREATE TABLE participates (
  p_id int REFERENCES participants,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (p_id, transcript_id)
);

CREATE TABLE mentions (
  location_id int REFERENCES locations,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (location_id, transcript_id)
);

CREATE TABLE describes (
  k_id int REFERENCES keywords,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (k_id, transcript_id)
);

CREATE VIEW participant_transcript_view AS
   SELECT * FROM (SELECT participants.*, transcripts.* FROM participates
                LEFT OUTER JOIN participants ON participants.p_id = participates.p_id
                LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
participates.transcript_id) as PTV;

CREATE VIEW location_transcript_view AS
   SELECT * FROM (SELECT locations.*, transcripts.* FROM mentions
                LEFT OUTER JOIN locations ON locations.location_id = mentions.location_id
                LEFT OUTER JOIN transcripts ON transcripts.transcript_id = mentions.transcript_id)
as LTV;

CREATE VIEW user_transcript_view AS
   SELECT * FROM (SELECT users.*, transcripts.* FROM bookmarks
                LEFT OUTER JOIN users ON users.user_id = bookmarks.user_id
                LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
bookmarks.transcript_id) as UTV;

CREATE VIEW keyword_transcript_view AS
   SELECT * FROM (SELECT keywords.*, transcripts.* FROM describes
                LEFT OUTER JOIN keywords ON keywords.k_id = describes.k_id
```

```
        LEFT OUTER JOIN transcripts ON transcripts.transcript_id = describes.transcript_id)
as UTV;

 CREATE VIEW full_transcript_view AS
   SELECT * FROM (SELECT keyword_transcript_view.keyword, keyword_transcript_view.k_id,
location_transcript_view.location_id, location_transcript_view.street_name,
participant_transcript_view.* FROM participant_transcript_view
        LEFT OUTER JOIN keyword_transcript_view ON
keyword_transcript_view.transcript_id = participant_transcript_view.transcript_id
        LEFT OUTER JOIN location_transcript_view ON
location_transcript_view.transcript_id = participant_transcript_view.transcript_id) as FT;
```

INSERT Examples. NOTE: Variables are placeholders. They can be replaced with any acceptable values based on the schema.

```
INSERT INTO transcripts
VALUES (title, summary, audio_path, text_path, text);

INSERT INTO participants
VALUES (name);

INSERT INTO keywords
VALUES (keyword);

INSERT INTO locations
VALUES (street_name);

INSERT INTO participates
VALUES (participant_id, transcript_id);

INSERT INTO mentions
VALUES (location_id,transcript_id);

INSERT INTO describes
VALUES (k_id,transcript_id);

INSERT INTO bookmarks
VALUES (user_id, transcript_id);
```

User Filtering for id (needed to localize data per user)

```
SELECT user_id FROM users WHERE email = email AND password_hash = password_hash;
```

Transcript Filtering (search is a placeholder for the search string done by the user)

SELECT * FROM full_transcript_view WHERE title ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE text_content ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE summary ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE keyword = search;

SELECT * FROM full_transcript_view WHERE street_name ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE name ILIKE ‘%search%’;

Bookmarks Filtering (search is a placeholder for the search string done by the user and id is the currently signed in user)

SELECT * FROM user_transcript_view WHERE user_id = id AND title ILIKE ‘%search%’;

SELECT * FROM user_transcript_view WHERE user_id = id AND text_content ILIKE ‘%search%’;

SELECT * FROM user_transcript_view WHERE user_id = id AND summary ILIKE ‘%search%’;

Delete entries (let id be a possible value for the id columns)

DELETE FROM transcripts where transcript_id = id;

DELETE FROM locations where location_id = id;

DELETE FROM keywords where k_id = id;

DELETE FROM participants where p_id = id;

DELETE FROM participates where p_id = id;

DELETE FROM participates where transcript_id = id;

DELETE FROM describes where k_id = id;

DELETE FROM describes where transcript_id = id;

DELETE FROM mentions where location_id = id;

DELETE FROM mentions where transcript_id = id;

DELETE FROM bookmarks where transcript_id = id;

Update Transcript (Strings are arbitrary. Id is a possible transcript id)

```
Update transcripts
SET title='title',
SET text_content='lorem ipsum',
SET summary='summary'
WHERE transcript = id;
```

Copy CSV transcript data (File path would be updated to reflect transcript)

```
COPY transcripts(title, text_file_path, audio_file_path, summary, text_content)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_transcript.csv' DELIMITER '|' CSV
HEADER;

COPY participants(name)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_participants.csv' DELIMITER '|'
CSV HEADER;

COPY locations(street_name)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_locations.csv' DELIMITER '|' CSV
HEADER;

COPY keywords(keyword)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_keywords.csv' DELIMITER '|' CSV
HEADER;
```

Construction: Tables, Queries, and User Interface

The tables are as described above:

Users, Transcripts, Participants, Locations, Keywords, Participates, Describes, Mentions, and Bookmarks.

We have a set of views: Participant_transcript_view, Keyword_transcript_view, Location_transcript_view, User_transcript_view, and full_transcript_view

Queries:

setup.sql -

```
-- Sets up all the tables and views
-- Queries created as a group. File created by Thomas Orth. Reviewed by Matthew Van Soelen and Justin Pabon


-- Creates the transcripts table
-- Text_content is meant to be unique but since the UNIQUE constraint creates an index, it cannot store large amounts of text
CREATE TABLE transcripts (
  transcript_id SERIAL PRIMARY KEY,
  title text UNIQUE,
  summary text UNIQUE,
  audio_file_path text UNIQUE,
  text_file_path text UNIQUE,
  text_content text
);

-- Creates the users table
CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  email text UNIQUE,
  password_hash text UNIQUE,
  lflag int
);

-- Creates the participants table
CREATE TABLE participants (
  p_id SERIAL PRIMARY KEY,
  name text UNIQUE
);

-- Creates the locations table
CREATE TABLE locations (
  location_id SERIAL PRIMARY KEY,
  street_name text UNIQUE
);

-- Creates the keywords table
CREATE TABLE keywords (
  k_id SERIAL PRIMARY KEY,
  keyword text UNIQUE
);

-- Creates the bookmarks table
```

```
CREATE TABLE bookmarks (
  user_id int REFERENCES users,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (user_id, transcript_id)
);

-- Creates the participates table
CREATE TABLE participates (
  p_id int REFERENCES participants,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (p_id, transcript_id)
);

-- Creates the mentions table
CREATE TABLE mentions (
  location_id int REFERENCES locations,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (location_id, transcript_id)
);

-- Creates the describes table
CREATE TABLE describes (
  k_id int REFERENCES keywords,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (k_id, transcript_id)
);

-- Creates a view that joins participants and transcripts
-- Its a representation of the participates relationship
CREATE VIEW participant_transcript_view AS
   SELECT * FROM (SELECT participants.*, transcripts.* FROM participates
                LEFT OUTER JOIN participants ON participants.p_id = participates.p_id
                LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
participates.transcript_id) as PTV;

-- Creates a view that joins locations and transcripts
-- Its a representation of the mentions relationship
CREATE VIEW location_transcript_view AS
   SELECT * FROM (SELECT locations.*, transcripts.* FROM mentions
                LEFT OUTER JOIN locations ON locations.location_id = mentions.location_id
                LEFT OUTER JOIN transcripts ON transcripts.transcript_id = mentions.transcript_id)
as LTV;

-- Creates a view that joins users and transcripts
```

```
-- Its a representation of the bookmarks relationship
CREATE VIEW user_transcript_view AS
  SELECT * FROM (SELECT users.*, transcripts.* FROM bookmarks
            LEFT OUTER JOIN users ON users.user_id = bookmarks.user_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
bookmarks.transcript_id) as UTV;

-- Creates a view that joins keywords and transcripts
-- Its a representation of the describes relationship
CREATE VIEW keyword_transcript_view AS
  SELECT * FROM (SELECT keywords.*, transcripts.* FROM describes
            LEFT OUTER JOIN keywords ON keywords.k_id = describes.k_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id = describes.transcript_id)
as UTV;

-- Creates a view that joins the mentions, describes and participates relationships into one view
 CREATE VIEW full_transcript_view AS
  SELECT * FROM (SELECT keyword_transcript_view.keyword, keyword_transcript_view.k_id,
location_transcript_view.location_id, location_transcript_view.street_name,
participant_transcript_view.* FROM participant_transcript_view
            LEFT OUTER JOIN keyword_transcript_view ON
keyword_transcript_view.transcript_id = participant_transcript_view.transcript_id
            LEFT OUTER JOIN location_transcript_view ON
location_transcript_view.transcript_id = participant_transcript_view.transcript_id) as FT;



-- Insert users

-- Dummy Password for the users are "password", "admin", and "lib"
INSERT INTO users(email, password_hash, lflag) VALUES ('test@gmail.com',
'$2b$12$B0ChMc2GW9RSKShJTqQnLOwemKAfQZPCPOG/pfQip8h6iTXGngQ9m', 0),
('test2@gmail.com',
'$2b$12$Ddna1VJdMgIjpcylmbsMjuhUDZb.c/oujD8S9gEOUuHaz9Z9Gm/92', 0),
('lib@gmail.com',
'$2b$12$bnWrAP.siJ.RNl1V5D66c.GZMXEu4rZGuWsuSpU8hmw/Ucm7L/uJ6', 1);

-- UPDATE WITH YOUR PATH ON YOUR SYSTEM FOR THE COPY COMMANDS

-- Insert first transcript and relevant information
COPY transcripts(title, text_file_path, audio_file_path, summary, text_content)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_transcript.csv' DELIMITER '|' CSV
HEADER;

COPY participants(name)
```

```
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_participants.csv' DELIMITER '|'
CSV HEADER;

COPY locations(street_name)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_locations.csv' DELIMITER '|' CSV
HEADER;

COPY keywords(keyword)
FROM '/home/lion/stage-v-group-1/data/joel/csv/insert_data_keywords.csv' DELIMITER '|' CSV
HEADER;

INSERT INTO participates VALUES (1,1);
INSERT INTO describes VALUES (1,1), (2,1);
INSERT INTO mentions VALUES (1,1), (2,1);
INSERT INTO bookmarks VALUES (1,1), (2,1);

-- Insert second transcript with relevant information
COPY transcripts(title, text_file_path, audio_file_path, summary, text_content)
FROM '/home/lion/stage-v-group-1/data/brenda/csv/insert_data_transcript.csv' DELIMITER '|'
CSV HEADER;

COPY participants(name)
FROM '/home/lion/stage-v-group-1/data/brenda/csv/insert_data_participants.csv' DELIMITER '|'
CSV HEADER;

COPY locations(street_name)
FROM '/home/lion/stage-v-group-1/data/brenda/csv/insert_data_locations.csv' DELIMITER '|'
CSV HEADER;

COPY keywords(keyword)
FROM '/home/lion/stage-v-group-1/data/brenda/csv/insert_data_keywords.csv' DELIMITER '|'
CSV HEADER;

INSERT INTO participates VALUES (2,2);
INSERT INTO describes VALUES (3,2), (4,2);
INSERT INTO mentions VALUES (3,2), (4,2);
INSERT INTO bookmarks VALUES (1,2);

-- Insert third transcript with relevant information
COPY transcripts(title, text_file_path, audio_file_path, summary, text_content)
FROM '/home/lion/stage-v-group-1/data/charles/csv/insert_data_transcript.csv' DELIMITER '|'
CSV HEADER;

COPY participants(name)
```

```
FROM '/home/lion/stage-v-group-1/data/charles/csv/insert_data_participants.csv' DELIMITER '|'
CSV HEADER;

COPY locations(street_name)
FROM '/home/lion/stage-v-group-1/data/charles/csv/insert_data_locations.csv' DELIMITER '|'
CSV HEADER;

COPY keywords(keyword)
FROM '/home/lion/stage-v-group-1/data/charles/csv/insert_data_keywords.csv' DELIMITER '|'
CSV HEADER;

INSERT INTO participates VALUES (3,3);
INSERT INTO describes VALUES (5,3), (6,3);
INSERT INTO mentions VALUES (4,3), (5,3);
```

Delete sql files (ids are demonstrative):

```
DELETE FROM bookmarks where transcript_id = 2;
DELETE FROM describes where transcript_id = 2;
DELETE FROM mentions where transcript_id = 2;
DELETE FROM participates where transcript_id = 2;

DELETE FROM bookmarks where user_id = 2;
DELETE FROM describes where k_id = 2;
DELETE FROM mentions where location_id = 2;
DELETE FROM participates where p_id = 2;

DELETE FROM keywords where k_id = 2;
DELETE FROM locations where location_id = 2;
DELETE FROM participants where p_id = 2;
DELETE FROM transcripts where transcript_id = 2;
```

Insert queries (variable placeholders for the tuples after VALUES)

```
INSERT INTO transcripts(title, summary, audio_file_path, text_file_path, text_content)
VALUES (t_title, t_summary, t_audio_path, t_text_path, t_text);

INSERT INTO participants(name)
VALUES (p_name);

INSERT INTO keywords(keyword
VALUES (k_keyword);

INSERT INTO locations(street_name)
```

VALUES (l_street_name);

INSERT INTO participates
VALUES (participant_id, transcript_id);

INSERT INTO mentions
VALUES (location_id,transcript_id);

INSERT INTO describes
VALUES (k_id,transcript_id);

INSERT INTO bookmarks
VALUES (user_id, transcript_id);

Filtering and Update - No change from Design portion. Refer to Elaboration: Design for more information

Additional queries (values are demonstrative):

SELECT COUNT(transcript_id) FROM user_transcript_view WHERE user_id=2 AND transcript_id=2;

SELECT COUNT(user_id) FROM users WHERE email='test@gmail.com';

SELECT COUNT(transcript_id) FROM bookmarks WHERE user_id=2 AND transcript_id=2;

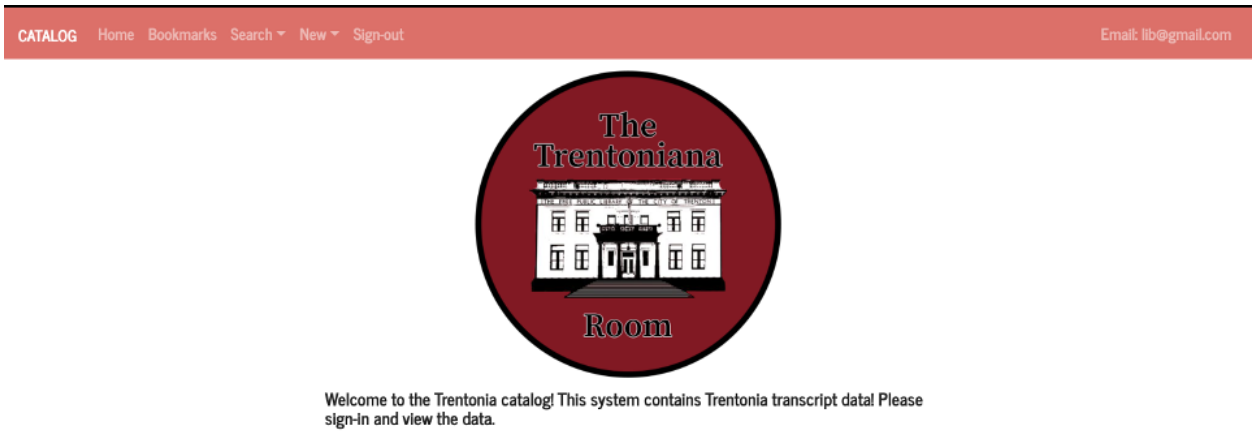
SELECT * FROM participants where name='Joe';

SELECT * FROM keywords where keyword='word';

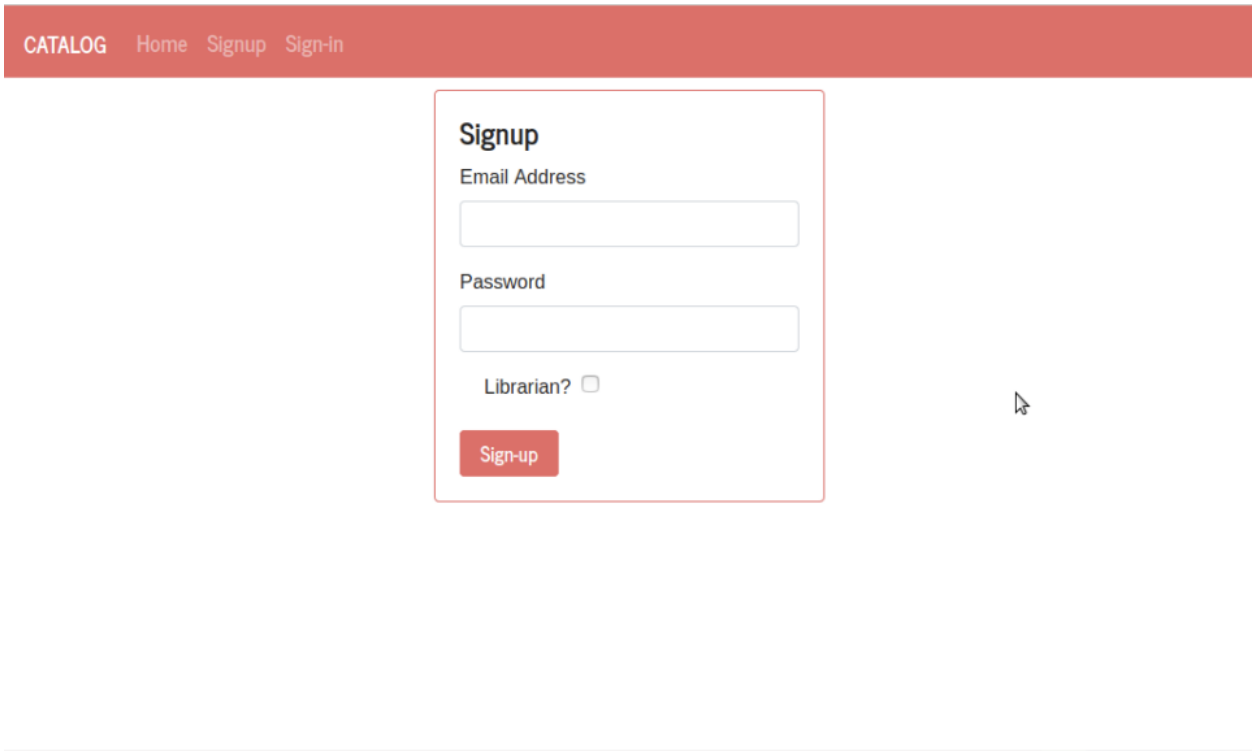
SELECT * FROM locations where street_name = 'main street';

Interface pages:

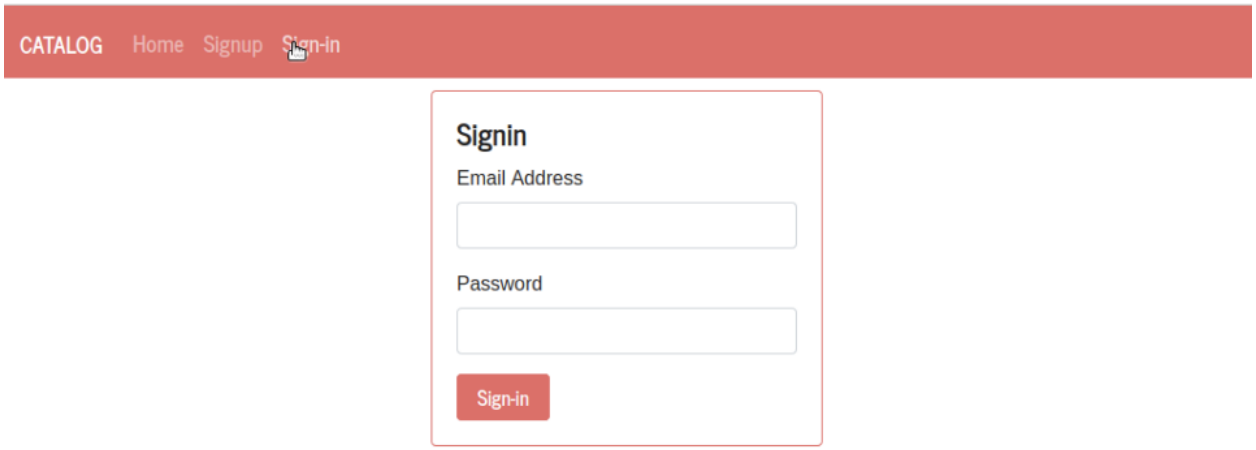
Main page:



Sign up:

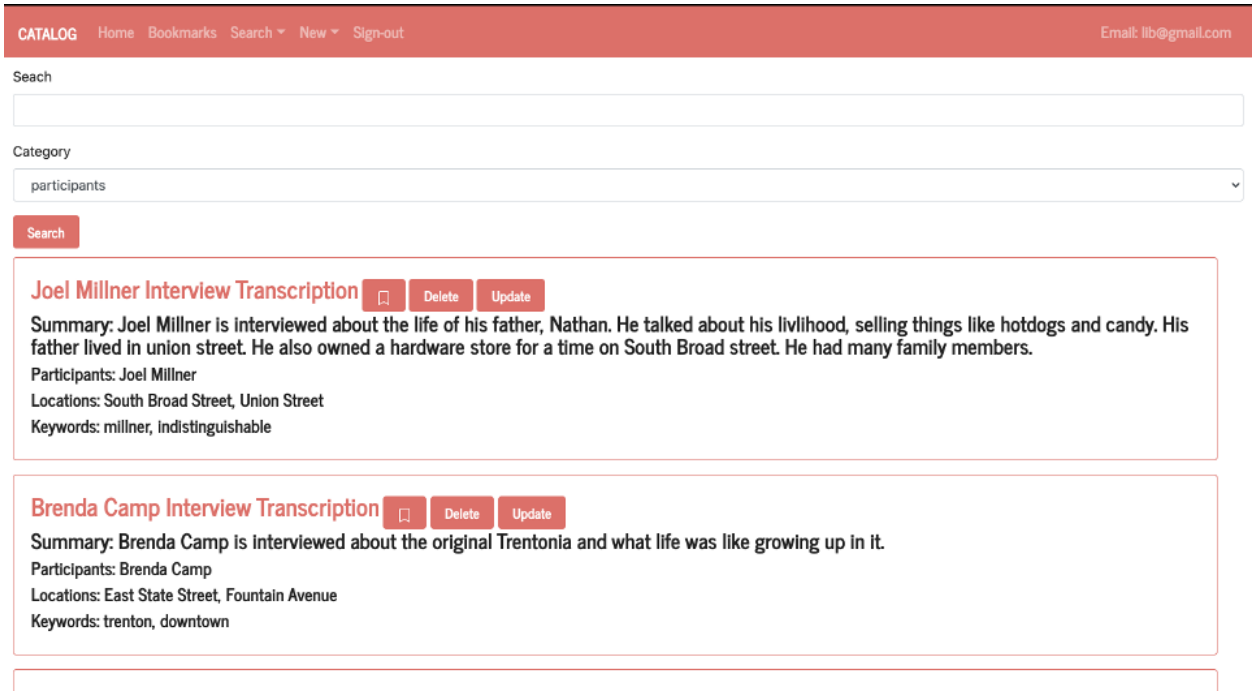

CATALOG
Home
Signup
Sign-in
Signup
Email Address
Password
Librarian?
Sign-up

Sign in:

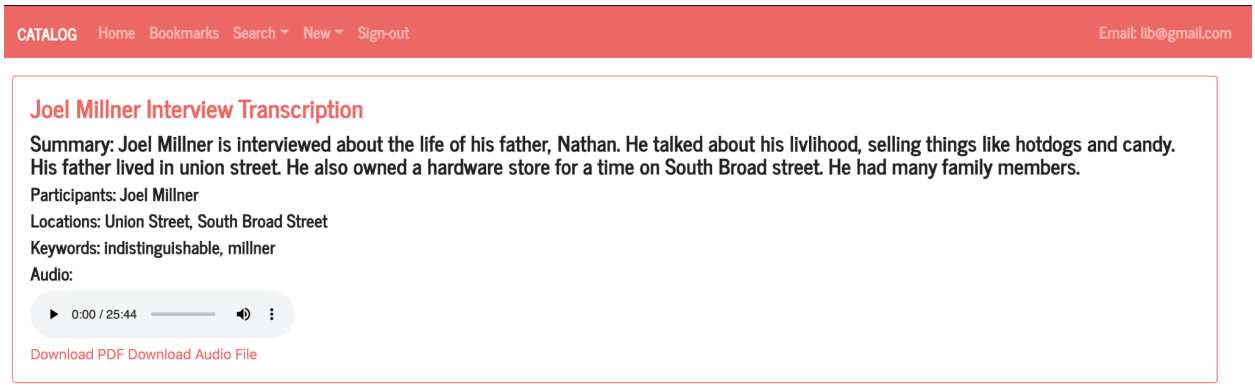


127.0.0.1:5000/auth/signin

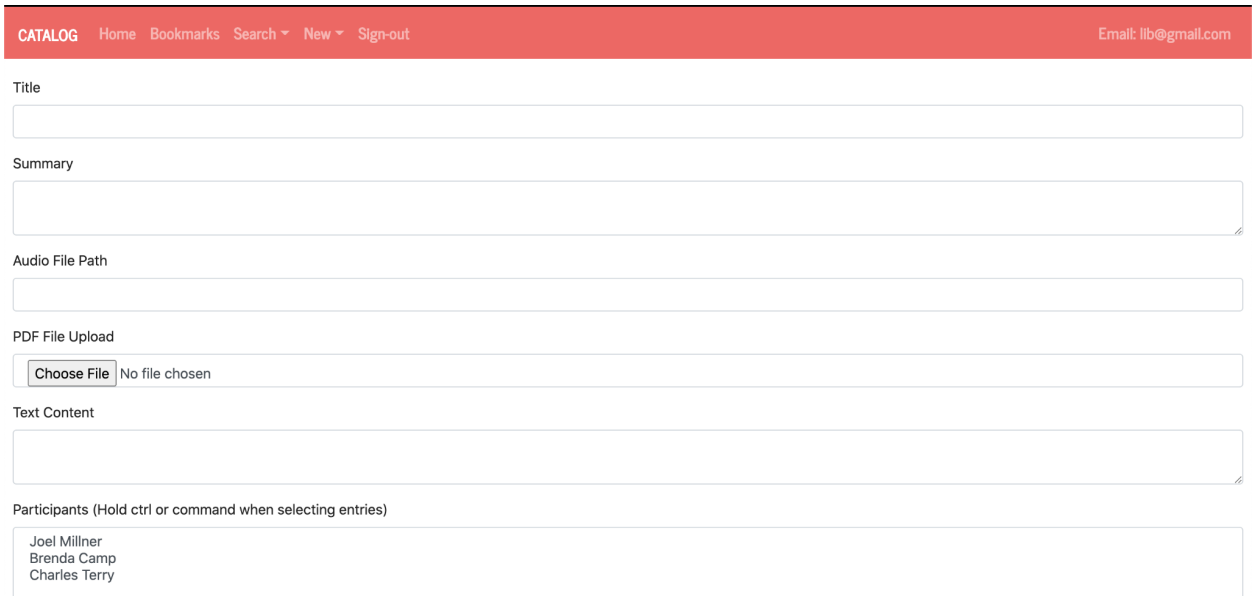
Transcript filtering



Transcript viewing



Transcript creation



Participants (Hold ctrl or command when selecting entries)

Joel Millner
Brenda Camp
Charles Terry

Keywords (Hold ctrl or command when selecting entries)

millner
indistinguishable
downtown
trenton

Locations (Hold ctrl or command when selecting entries)

Union Street
South Broad Street
Fountain Avenue
East State Street

Submit

Transcript update

CATALOG Home Bookmarks Search New Sign-out Email: lib@gmail.com

Title

Joel Millner Interview Transcription

Summary

Joel Millner is interviewed about the life of his father, Nathan. He talked about his livlihood, selling things like hotdogs and candy. His father lived in union street. He also owned a hardware store for a time on South Broad street. He had many family members.

Audio File Path

https://archive.org/download/JHS24SideB/JHS%2024-%20side%20B.mp3

PDF File Upload

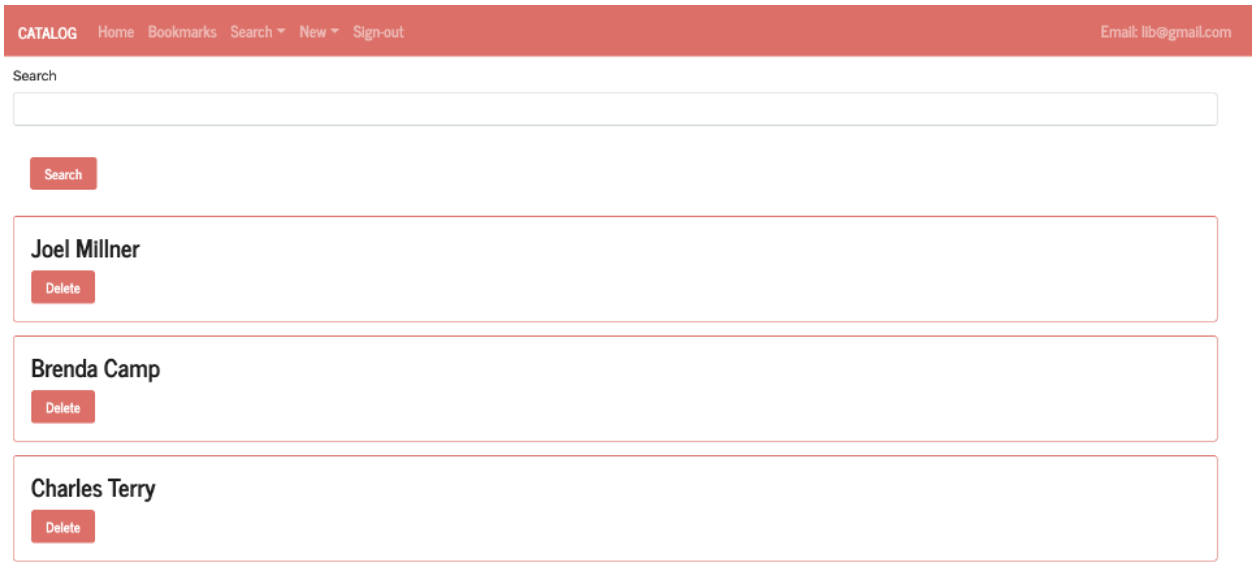
Choose File No file chosen

Text Content

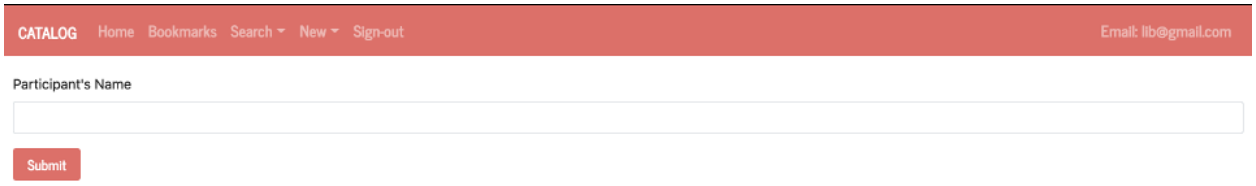
INTERVIEWER: Today is, uh, May the 31st 1995, I'm sitting with, uh, Joel Millner in his spacious office, and uh we're gonna explore a little more information of his dad Nathan Millner, which I recall him very well, uh, a few years back. Uh, the first thing, uh, J I wanna do is to clarify, uh, this first date when he was born. What year was he born? It's—I had somebody mention

Submit

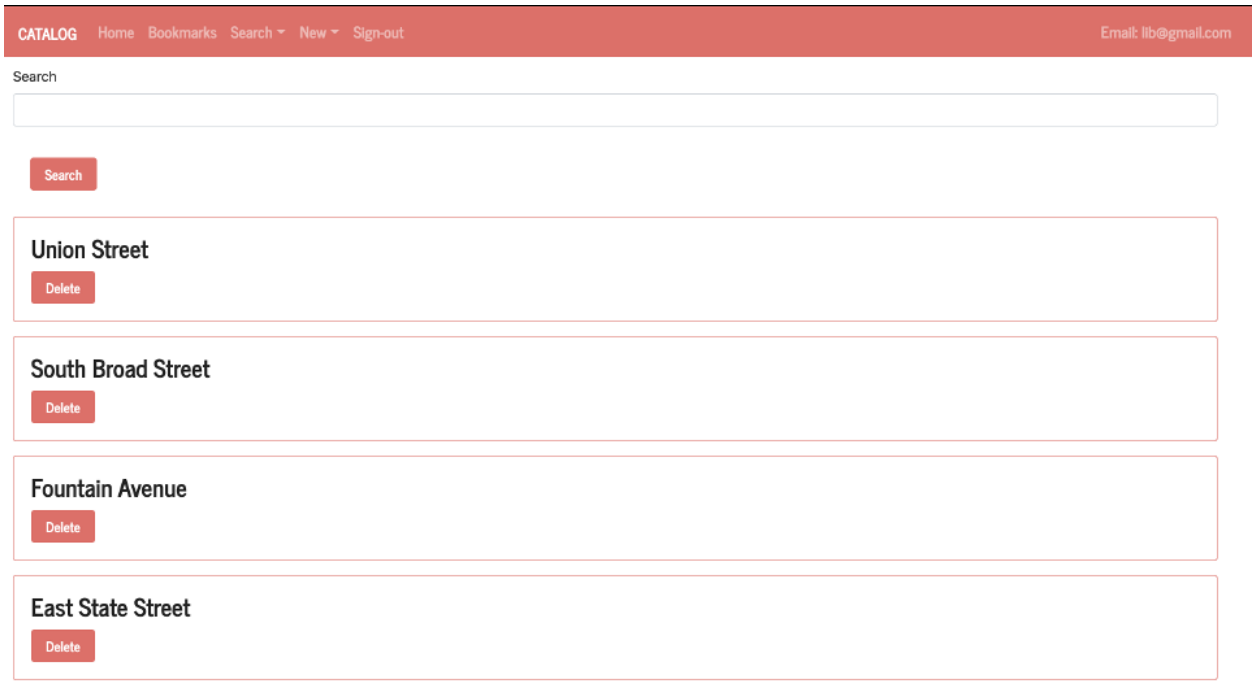
Participants filtering:



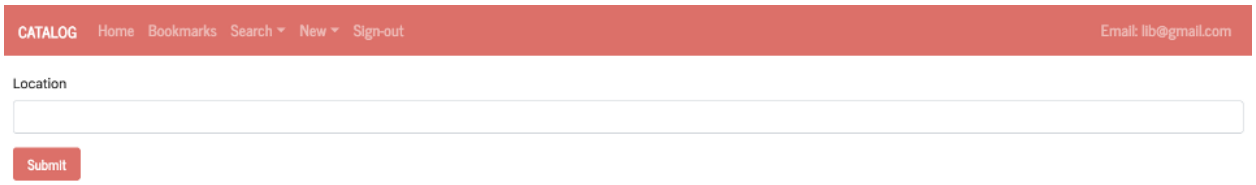
Participants creation:



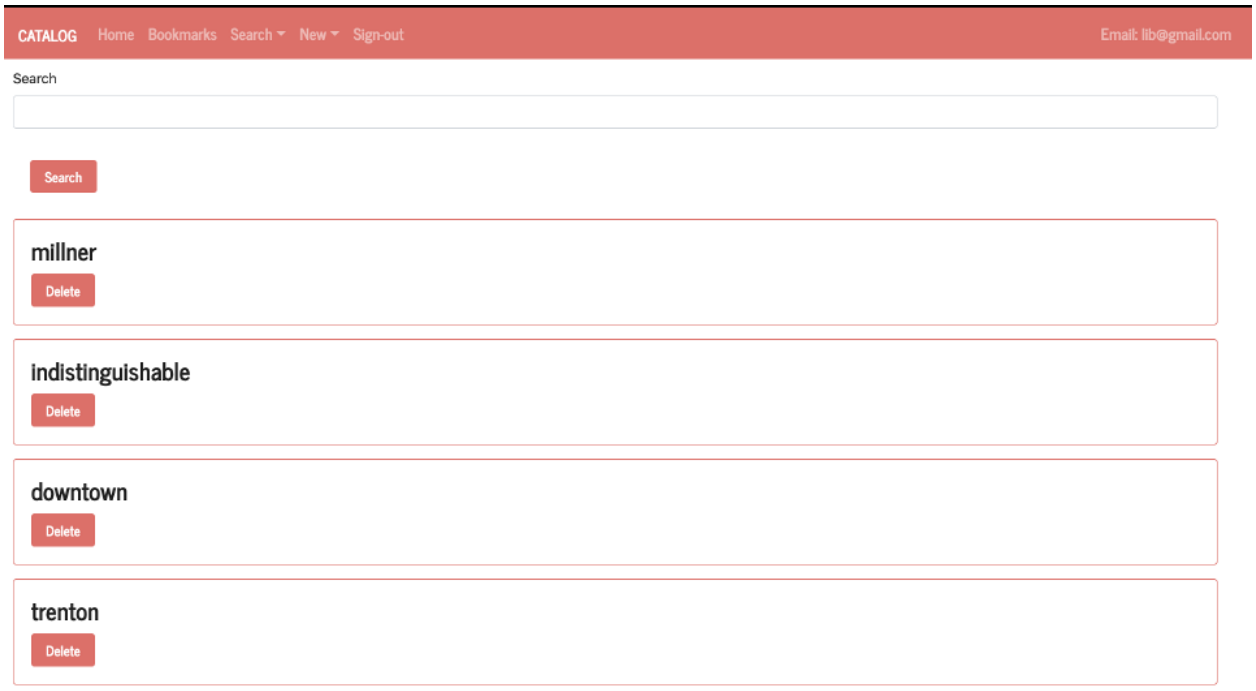
Locations filtering:



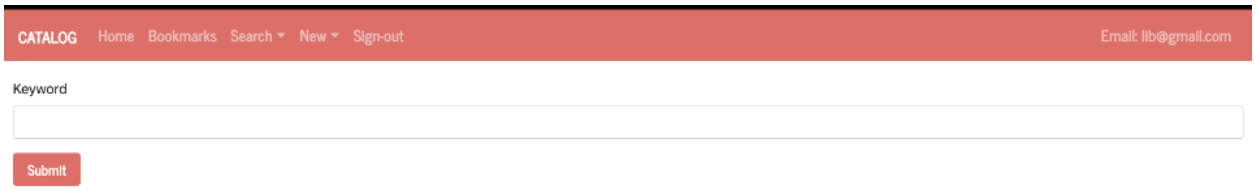
Locations creation:



Keyword filtering:



Keyword creation:



Transition: Maintenance

Code is appropriately organized for each python file with maintenance information and explanation of logic

Transition: Product Hand Over

Project made public and accessible via the link:
https://github.com/TomOrth/trenton-catalog-system