Team 4

Project Name: Trentaudio

Gabrielle Curcio, curciog1@tcnj.edu, Github: Gabscurioshop, (732) 279-5021

Carolyne Ulm, ulmc1@tcnj.edu, Github: CarolyneH, (908) 752-7943

Xuanyi Zhao, zhaox2@tcnj.edu, Github: ZZZXXX0110, (516) 439-3439

Inception: "Executive Summary"

Need:

- Needs a user-interface that's easy to navigate, easy to search in the database, and quick to retrieve search results
- Needs to search by keywords that describe the content
- Needs to see what each file is about without having to read/listen to the entire file
- Security so files can't be changed without permission

Approach

- Rely on user comments to classify metadata about the files
- Rely on users to report errors in the transcripts (that admin reviews)
- List metadata for file in the search results
- Show transcript together with audio file
- Grant administrator access to all database features
- Only allow admin to add/change/edit files

Benefits

- Easier-to-navigate database will be more likely to be used by people doing research and will receive more internet traffic from it
- The librarian will not have as large a workload with reading/listening to files in order to find errors and meta-data
- Simpler, more efficient database will be easier for them to manage

Cost

- Our user reporting error and metadata submission system is unique to our project and helps relieve workload on admin- not present in other systems
- Our system is a new website separate from Trentoniana, so they would be required to switch over to a new system

Elaboration: Project Proposal and Specifications

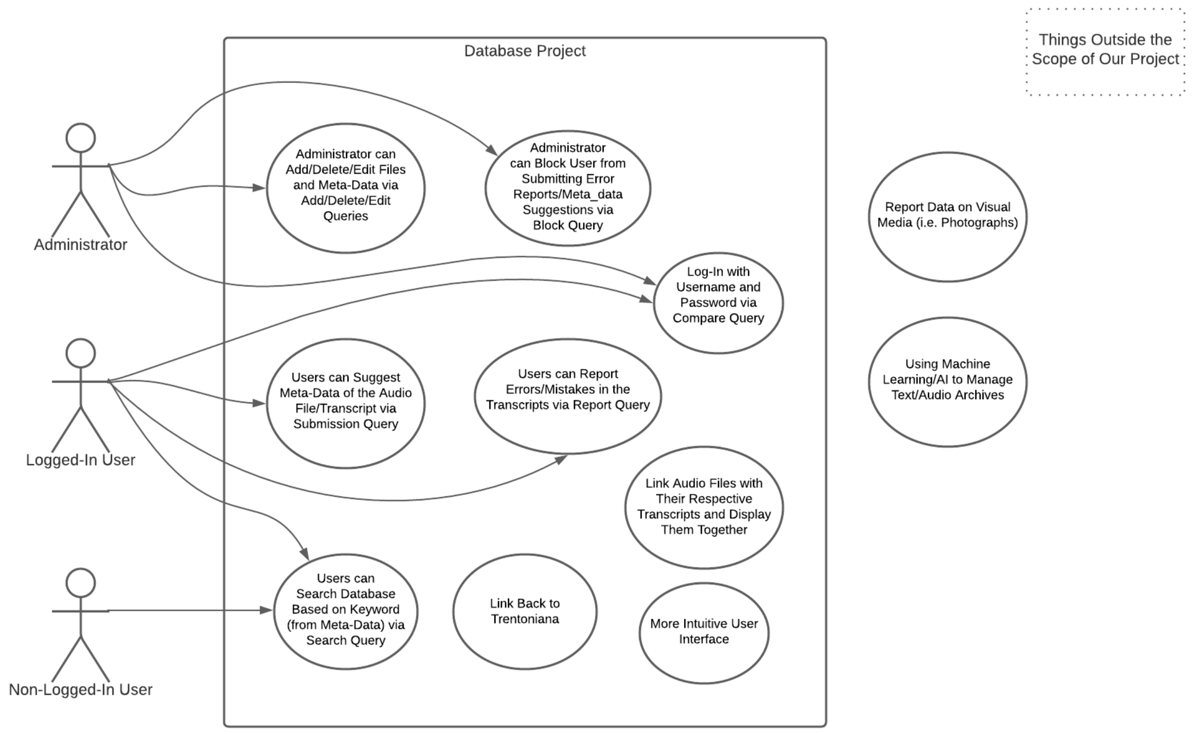
Project Proposal:

- **Problem statement:**
    - In the Trentonian Library online, audio files are stored inefficiently, making it difficult for users to search and sort through the files to find those that are best suited for their needs. Additionally, the user interface is overly complicated and confusing for a casual user. These problems with the library must be addressed to improve the user's experience.
- **Objective of the module:**
    - Creating a clearer user interface that allows for better access to the data (easier to use search and sort functions).
    - Creating a database system that handles audio-based data.
    - Link audio files to their corresponding transcripts.
    - Allow upper-level users to submit possible meta-data for files (such as topics addressed in the file) or issues/errors found with the file.
    - Allow casual-level users (such as people not logged on) to view transcripts and listen to audio files (presented on the same page).
    - Allow an administrator to edit meta-data or add/delete/edit files.
- **Description of the desired end product, and the part you will develop for this class:**
    - A database system, including a user interface, that allows better access and management of the Trentonian Library archive.
    - This database will contain different levels of permissions/operations to be done based upon who the user is.
    - Certain users will be able to submit issues or errors they found with the transcript and they can submit meta-data about the file (that could make the file more easily searchable) and the administrator can review the submission to see if they should change the file.
- **Description of the importance and need for the module, and how it addresses the problem.**
    - Currently, the audio and visual content on the Trentonian Library provides limited metadata and no transcriptions on certain content.
    - We would like to expand the versatility of the Trentonian Library and make it a better tool for research.
    - Allowing for users to submit error fixes or meta-data would reduce the amount of work a single person (like a librarian) would have to do to maintain the database.
    - We will make the user interface more intuitive and allow for easier searching of files with the inclusion of more meta-data, making it simpler for someone researching to find what they need (for example- they want an audio file that talk

about moving costs, that could be a keyword they could search by in our database, which the Trentonian Library doesn't have).

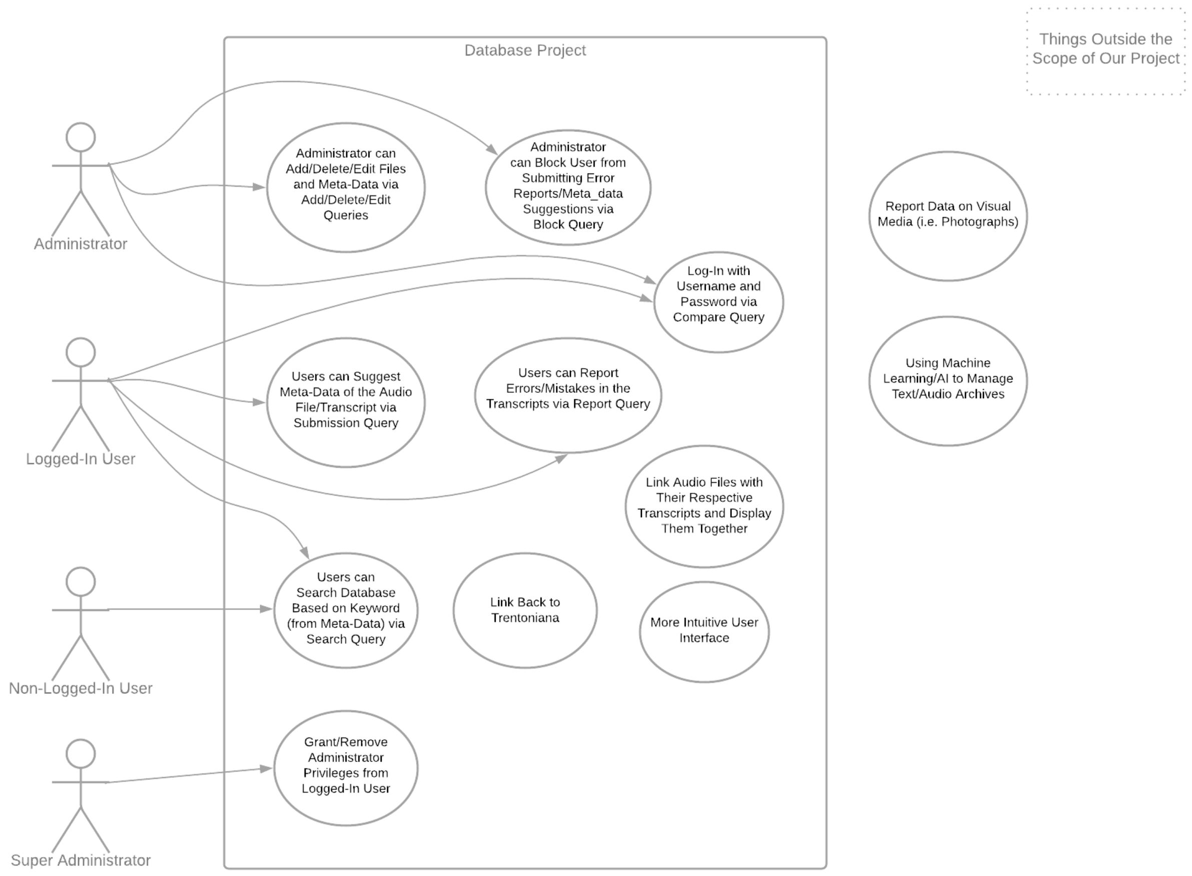
- **Plan for how you will research the problem domain and obtain the data needed.**
    - We will look at other audio storage systems (like other online libraries) to decide how we can best set up our user interface.
    - Look at ways how transcript files are typically stored.
    - Our data will come from the Trentonian Library and Dr. Steele's transcripts.
    - Some data, as mentioned, will come from users sending in edits/meta-data suggestions and we could try out this functionality during the testing phase.
- **Other similar systems/ approaches that exist, and how your module is different or will add to the existing system.**
    - Garland Encyclopedia of World Music Online: When a user finds a sound recording or a video, they can play audio while reading some metadata about the recording. An interesting feature about this library is it displays accompanying transcripts to videos.
    - Schomburg Center for Research in Black Culture: When a user clicks on a sound recording entry, they will be taken to a page with the original audio and some metadata (title, date, keywords, etc). However, the page does not display the transcript for the interview.
    - The Library of Congress stores audio files with their metadata like interviewer name, date, and file topic but it doesn't show the transcripts for the files. It also allows a user to search files like our system does.
    - The Southern Oral History Project (https://dc.lib.unc.edu/cdm/compoundobject/collection/sohp/id/12031/rec/1) has a good search feature and allows users to sort records by certain fields. The audio files are stored with metadata (visible to the user) like the description of the file, interviewer name/date of birth, interviewer ethnicity, and citation for the record (which is a good idea to add to ours). They also show the transcript with the audio file.
    - While the libraries mentioned above have good user interfaces, ours improves upon them with the user suggestion function, thereby providing help to the librarian with file management.
- **Possible other applications of the system (how it could be modified and reused.)**
    - The system is a system that allows indexing of any data with machine-readable content, and can be potentially used for managing other text/audio archives.
    - The system could be updated to allow for visual media such as the photographs in the Trentonian Library (allowing for people to more easily make a report of who they believe the person in the picture is or submitting descriptions of the context of the photo/video).

    - Implement AI/Machine Learning to analyze the transcript to extract meta-data quickly and without a lot of work on the part of the librarians/database managers.
- **Performance –specify how and to what extent you will address this.**
    - Controlled redundancy: using repeated fields in a database. This helps improve query performance and increase database access speed
    - Denormalization: To avoid searching through multiple files to collect data
    - Transcripts are large files that must be stored in an efficient manner, so we will need to store it and retrieve them in such a way that can handle the large amounts of data.
- **Security –specify how and to what extent you will provide security features.**
    - Only allow the administrator to have permission to change/add/delete the actual files.
    - Have encoded passwords so that people cannot get easy access to the administrator's account.
    - If time allows, have a function to allow the administrator to suspend a user if they submit too many error/meta-data suggestions in order to avoid spamming.
    - Assign stricter roles (like people have to be signed in to make error/meta-data suggestions).
    - Use parameterized queries and use constraints to prevent SQL injections.
- **Backup and recovery –specify how and to what extent you will implement this.**
    - Transaction log backup: save all transaction log entries and transaction events that occurred in the database
    - Differential backup: Saves all data since last backup; Perform a daily data backup routine
    - If part of a transaction is not completed, revert the database back to the state before the incomplete transaction began.
- **Technologies and database concepts the team will need to learn, and a plan for learning these.**
    - Build upon current knowledge of Postgres- learn by reading Chapter 2 of Seven Databases in Seven Weeks book and online tutorials
    - Programming language that interfaces with Postgres: learn by using online tutorials- possibly .Net, C, C++, JavaScript, Node.js
- **A diagrammatic representation of the system boundary that specifies what data you will model and which queries you will implement.**



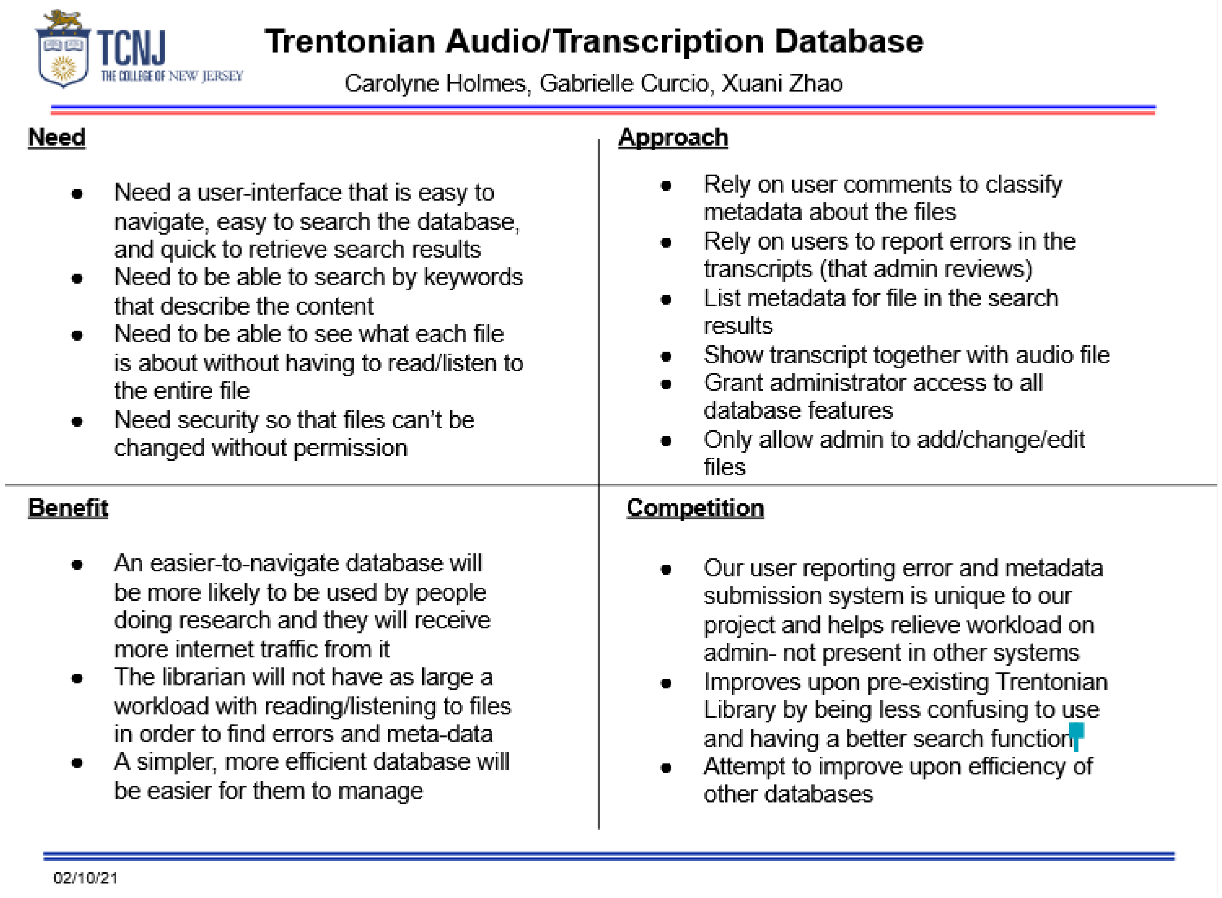
Database Project
Things Outside the Scope of Our Project
Administrator
Logged-In User
Non-Logged-In User
Administrator can Add/Delete/Edit Files and Meta-Data via Add/Delete/Edit Queries
Administrator can Block User from Submitting Error Reports/Meta_data Suggestions via Block Query
Log-In with Username and Password via Compare Query
Users can Suggest Meta-Data of the Audio File/Transcript via Submission Query
Users can Report Errors/Mistakes in the Transcripts via Report Query
Link Audio Files with Their Respective Transcripts and Display Them Together
Users can Search Database Based on Keyword (from Meta-Data) via Search Query
Link Back to Trentoniana
More Intuitive User Interface
Report Data on Visual Media (i.e. Photographs)
Using Machine Learning/AI to Manage Text/Audio Archives

- We edited this diagram by adding another role- the Super Administrator that can grant and remove administrator privileges/role from logged-in users


Database Project
Things Outside the Scope of Our Project
Administrator
Logged-In User
Non-Logged-In User
Super Administrator
Administrator can Add/Delete/Edit Files and Meta-Data via Add/Delete/Edit Queries
Administrator can Block User from Submitting Error Reports/Meta_data Suggestions via Block Query
Log-In with Username and Password via Compare Query
Users can Suggest Meta-Data of the Audio File/Transcript via Submission Query
Users can Report Errors/Mistakes in the Transcripts via Report Query
Link Audio Files with Their Respective Transcripts and Display Them Together
Users can Search Database Based on Keyword (from Meta-Data) via Search Query
Link Back to Trentoniana
More Intuitive User Interface
Grant/Remove Administrator Privileges from Logged-In User
Report Data on Visual Media (i.e. Photographs)
Using Machine Learning/AI to Manage Text/Audio Archives

  ○

- **1-page quad chart; see: Quad_instructions_template.ppt in the Canvas files section**



Trentonian Audio/Transcription Database

Carolyne Holmes, Gabrielle Curcio, Xuani Zhao

Need

- Need a user-interface that is easy to navigate, easy to search the database, and quick to retrieve search results
- Need to be able to search by keywords that describe the content
- Need to be able to see what each file is about without having to read/listen to the entire file
- Need security so that files can't be changed without permission

Approach

- Rely on user comments to classify metadata about the files
- Rely on users to report errors in the transcripts (that admin reviews)
- List metadata for file in the search results
- Show transcript together with audio file
- Grant administrator access to all database features
- Only allow admin to add/change/edit files

Benefit

- An easier-to-navigate database will be more likely to be used by people doing research and they will receive more internet traffic from it
- The librarian will not have as large a workload with reading/listening to files in order to find errors and meta-data
- A simpler, more efficient database will be easier for them to manage

Competition

- Our user reporting error and metadata submission system is unique to our project and helps relieve workload on admin- not present in other systems
- Improves upon pre-existing Trentonian Library by being less confusing to use and having a better search function
- Attempt to improve upon efficiency of other databases

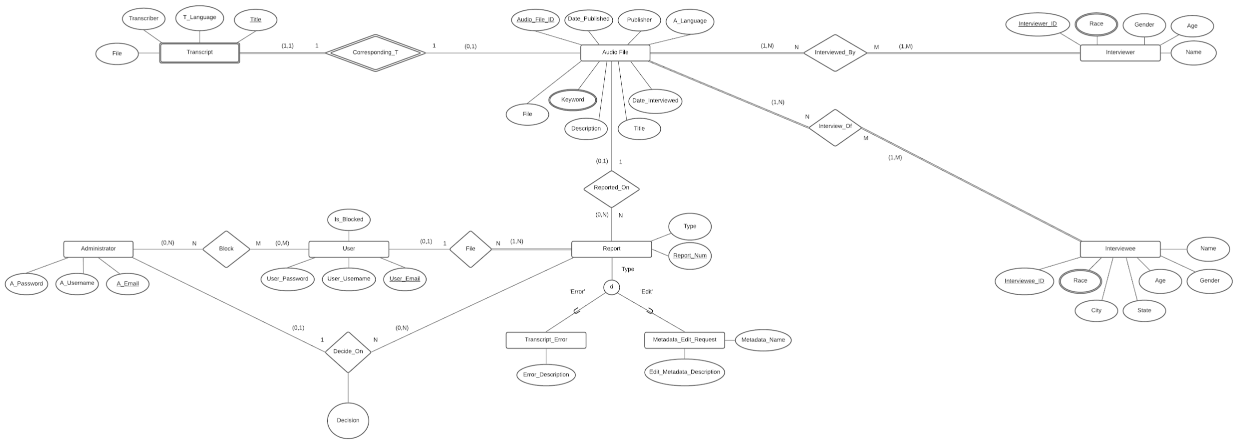
02/10/21

○

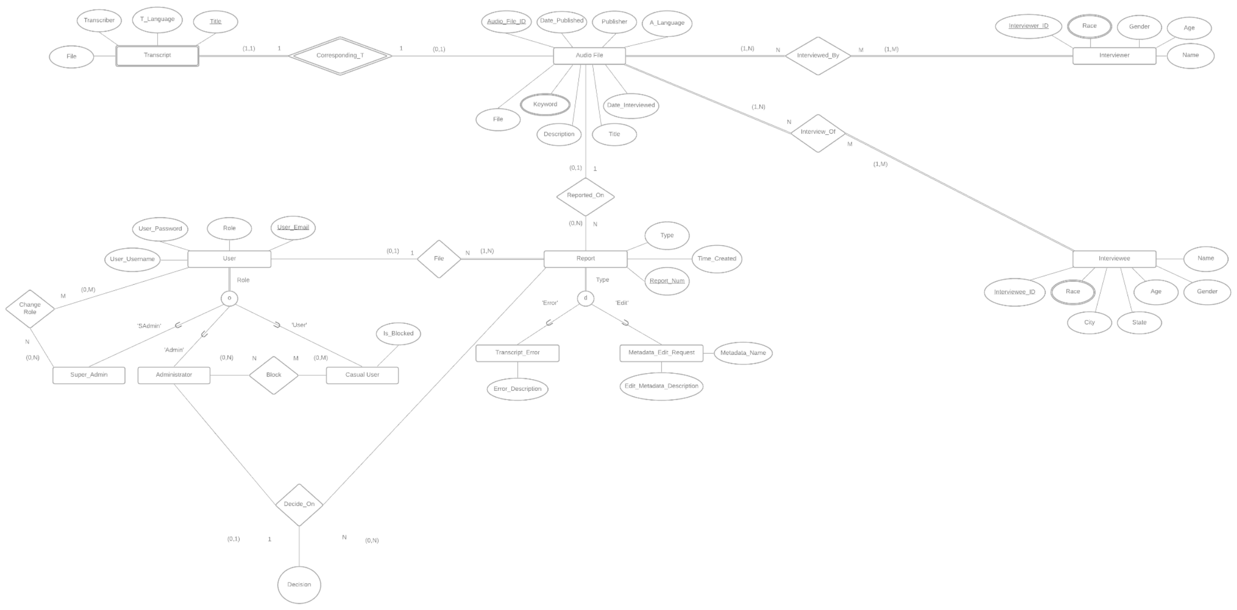
Elaboration: Design: Stage III and IV docs

Stage III Database Model:

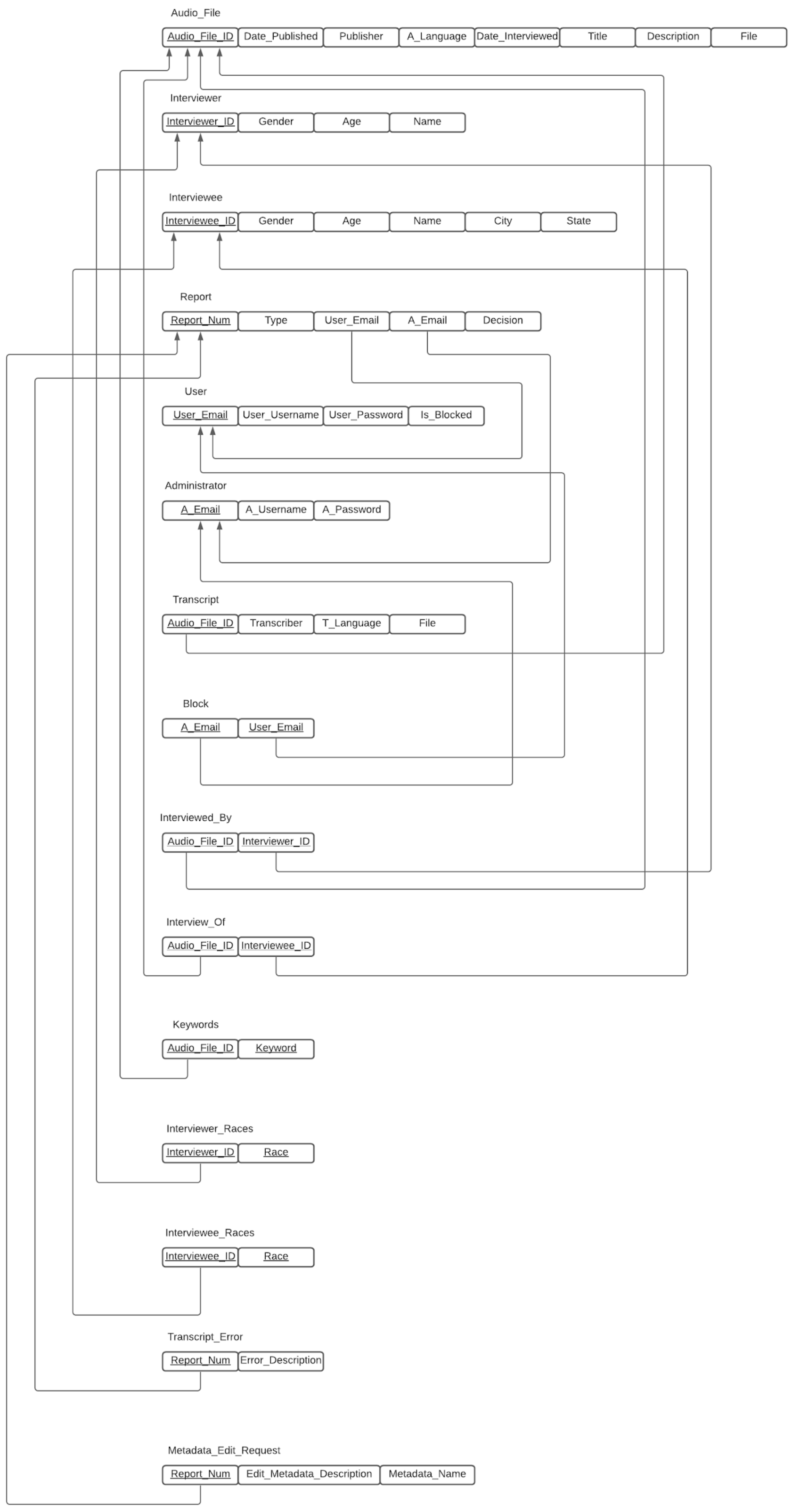
EER Model:
https://lucid.app/lucidchart/8af1a69d-400a-44f8-82c0-9317842e2c2a/edit?page=0_0#



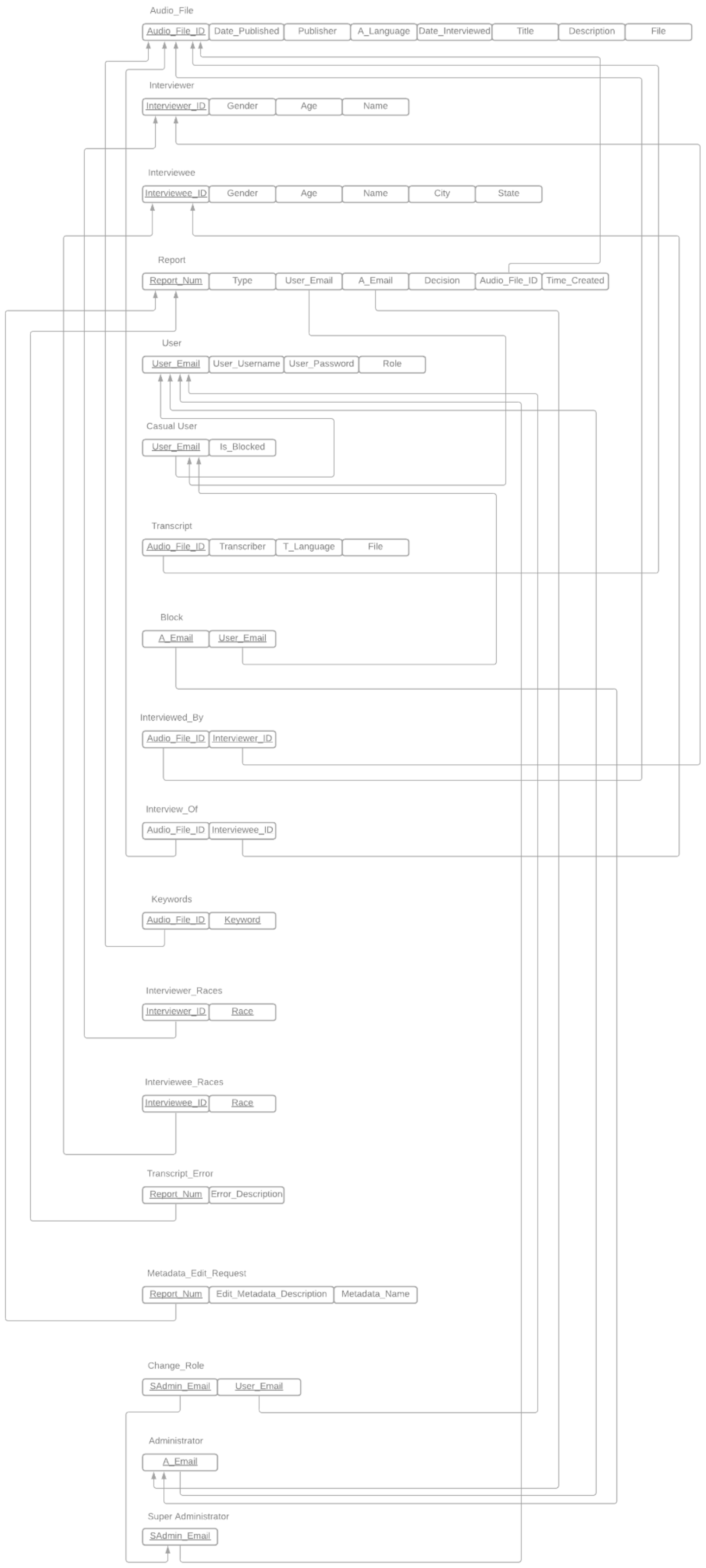
We edited this (as seen below) by adding a time_created attribute for reports and we added the Super Administrator entity that can change admin privileges for users and administrators.
We also adjusted it so that all of the types of roles were the specializations of the user class since they all shared similar attributes.



Relational Diagram:
https://lucid.app/lucidchart/8af1a69d-400a-44f8-82c0-9317842e2c2a/edit?page=sBHjS.yifsY2#



We edited this for the final stages in the same ways as mentioned for the EER diagram. We also added the attribute of Audio_File_ID for each report.


Audio_File
Audio_File_ID | Date_Published | Publisher | A_Language | Date_Interviewed | Title | Description | File
Interviewer
Interviewer_ID | Gender | Age | Name
Interviewee
Interviewee_ID | Gender | Age | Name | City | State
Report
Report_Num | Type | User_Email | A_Email | Decision | Audio_File_ID | Time_Created
User
User_Email | User_Username | User_Password | Role
Casual User
User_Email | Is_Blocked
Transcript
Audio_File_ID | Transcriber | T_Language | File
Block
A_Email | User_Email
Interviewed_By
Audio_File_ID | Interviewer_ID
Interview_Of
Audio_File_ID | Interviewee_ID
Keywords
Audio_File_ID | Keyword
Interviewer_Races
Interviewer_ID | Race
Interviewee_Races
Interviewee_ID | Race
Transcript_Error
Report_Num | Error_Description
Metadata_Edit_Request
Report_Num | Edit_Metadata_Description | Metadata_Name
Change_Role
SAdmin_Email | User_Email
Administrator
A_Email
Super Administrator
SAdmin_Email

Updated Diagrams (before the Final Stages):
Removed "Interviews Relationship" between Interviewer and Interviewee to reduce Interviewer/Interviewee redundancy
Removed "Browse" Relationship between User and Audio_File since non-logged in users can browse files.
Added File Attributes to Audio_File and Transcript to represent the raw file.

Database Initial Size= 200 audio records
Based upon size of trentoniana (74 audio records) and other databases, which are typically larger.

Types of Queries= SELECT by keyword (for searches), PROJECT specific data (like certain metadata for the files), SORT files by things like date or alphabetical order, ADD/DELETE/UPDATE record by Administrator, & submission of changes/errors by users

Numbers of Queries= likely 10-50 searches a day (SELECT and PROJECT and SORT operations), and likely initially many adding/updating records (but then only around 5 add/delete/update record operations every few weeks since the audio files would not change a lot and not many users would frequently submit metadata changes or transcript errors)

Stage IV
We slightly edited our diagrams to remove some redundancy (like title was in transcript and audio file and the browse relation wasn't really needed)

Demonstrate that all the relations in the relational schema are normalized to Boyce–Codd normal form (BCNF).
All of the diagrams follow the requirements: 1NF - all non-prime attributes are functionally dependent on another attribute and there are no nested tables, 2NF - there are no partial functional dependencies, 3NF - there are no transitive functional dependencies, and BCNF - there are no functional dependencies where the key of the relation is dependent on another attribute. To see the diagram for the normalizations, see the end of the document.

Audio_file: BCNF

    Audio_File_ID -> Date_Published
    Audio_File_ID -> Publisher
    Audio_File_ID -> A_Language
    Audio_File_ID -> Date_Interviewed
    Audio_File_ID -> Title
    Audio_File_ID -> Description
    Audio_File_ID -> File

Interviewer: BCNF

    Interviewer_ID -> Gender
    Interviewer_ID -> Age
    Interviewer_ID -> Name
    Interviewee: BCNF
    Interviewee_ID -> Gender
    Interviewee_ID -> Age
    Interviewee_ID -> Name
    Interviewee_ID -> City
    Interviewee_ID -> State
    We consider City and State to not be functionally dependent since a city name can belong to multiple states and a state can have multiple city values, so neither can predict a unique value for the other

Report: BCNF

    Report_Num -> Type
    Report_Num -> User_Email
    Report_Num -> A_Email
    Report_Num -> Decision

User: BCNF

    User_Email -> User_Username
    User_Email -> User_Password

User_Email -> Is_Blocked

Administrator: BCNF

A_Email -> A_Username

A_Email -> A_Password

Transcript: BCNF

Audio_File_ID ->Transcriber

Audio_File_ID ->T_Language

Audio_File_ID ->File

Block: BCNF

{A_Email,U_Email}

Interviewed_by: BCNF

Audio_File_ID->Interviewer_ID

Interview_of: BCNF

Audio_File_ID ->Interviewee_ID

Keywords: BCNF

{Audio_File_ID, Keyword}

Interviewer_Races: BCNF

{Interviewer_ID, Race}

Interviewee_Races: BCNF

{Interviewee_ID, Race}

Transcript_Error: BCNF

Report_Num -> Error_description

Metadata_Edit_Request: BCNF

Report_num -> Edit_metadata_description

Report_num -> Metadata_name

All tables were in BCNF, so not changes were made (besides those from question #1). Note- all tables are still in BCNF form, even with the changes made in the final stages.

Define the different views (virtual tables) required. For each view list the data and transaction requirements. Give a few examples of queries, in English, to illustrate.
Note- all final search results will be displayed in date_published order (newest to oldest)

Keyword Search View:

- Data = keywords entered by user
- Requirement = list audio file table information for each audio file that contains (or pattern matches) the keywords entered as display the audio file title, date published, and some lines of file description.
- Select from the keywords table, the audio file IDs of files that have the keyword (do this for each keyword entered sans words like ‘the’ or ‘a’ and order by best match). Take the

union of the results of each keyword search (from the temporary view created) to get all relevant audio file IDs (or select via pattern matching where the keywords are OR'd together), then join the union with the audio files table to get the information of all audio files with those keywords. We may also do something along the lines of finding files that contain more of the keywords than others (like a file that has 'blue whale' and a file that has 'blue' so for the search 'big blue whale', the first file will appear first) by finding the intersection of the results of the keyword searches.

- Example = select audio_ID where keywords = 'blue' OR 'whale' OR 'big' from keywords

Interviewer Search View:

- Data = interviewer name from user search
- Requirements = display all audio files that have an interviewer with the name that was searched
- Select from the interviewer table all interviewer IDs that have a name matching that which was searched and create a temporary view of these results. Then join these interviewer IDs with the interview_by table to get the audio file IDs of all files by that interviewer and store in a view. Then join the audio IDs with the audio file table to get the information of all the relevant files, similarly to the previous table.
- Example = select interviewer_ID where name = 'Cindy Smith' from interviewer;
    - select audio_ID where interviewer_ID = searched_IDs from interview_by
    - Select audio information from audio files where interviewer is Cindy Smith

Interviewee Search View:

- Data = interviewee name searched by user
- Requirements = display all audio files that have an interviewee with the name that was searched
- Select from the interviewee table all interviewee IDs that have a name matching that which was searched and create a temporary view of these results. Then join these interviewee IDs with the interview_of table to get the audio file IDs of all files of that interviewee and store in a view. Then join the audio IDs with the audio file table to get the information of all the relevant files, similarly to the previous table.
- Example = select interviewee_ID where name = 'Bob Ross' from interviewee;
    - select audio_ID where interviewee_ID = searched_IDs from interview_of
    - Select audio information from audio files where interviewee is Bob Ross

Race Search View:

- Data = race searched by user
- Requirements = display all audio files information (the same information displayed as in the other views) with interviewee race that matches what was searched; the race options are listed like checkboxes to prevent someone entering a race not found in the database
- Select the interviewee IDs that have a race matching what was searched from the interviewee_race table and save the results as a view. Then use a join (or optionally a

nested select) of the interviewee IDs with the interview_of table to get the audio file information of these interviewees and store the audio IDs in a view. Then use a join of these audio IDs and the audio file table to get the information of the relevant audio files.

- Example = select interviewee_ID where race = 'Native American' from interviewee_race;
    - select audio_ID where interviewee_ID = searched_IDS from interview_of
    - Select audio information from audio files where interviewee is Native American

City/State Search View:

- Data = city and/or state selected from a checkbox list by the user
- Requirements = display only the information about audio files in which the person being interviewed is from the searched city and/or state
- In the case of the city or state only search, select the interviewee ID from the interviewee tabel when the city or state matches what was searched. If both a city and state are searched, select the interviewee ID when city and state both match what was searched. Store this as a view and perform a join with the interview_of table to get the audio_IDs that apply to the given search city/state (this could also be done with the IN operation, it depends on what is most efficient). Store these results in a view and perform a join of the view with the audio file table to get the information about the relevant audio files.
- Example = select interviewee_ID where city = 'Trenton' AND state = 'NJ' from interviewee;
    - select audio_ID where interviewee_ID = searched_IDs from interview_of
    - Select audio information from audio files where interviewee is from Trenton, NJ

Reports View:

- Data = administrator decided to click a button to display reports and selected which type of report they would like to view; optionally, may allow administrator to view reports from a searched user (by username) or only users who have not been blocked, but we haven't decided yet.
- Requirements = display the reports, the username and email of the reporter, and part of the description of the report.
- Select from the reports table all reports that match the selected type and store the report numbers in a view. Use this view in a join, IN operation, etc. with the table that matched the report type (either metadata edit or transcript error) and then project the desired information from the report so that the administrator can see it.
- Example = select report_num where type = 'edit' from report;
    - join the reports view with metadata_edit table;
    - project user_email, user_username, description, and decision
    - Select report information from report where type is edit

Blocked Users View:

- Data = administrator clicks a button to view a list of blocked users, no other data is entered

- Requirements = show the username, user email, and administrator email or username who blocked the user
- Select from the list of users the user emails of those who have is_Blocked set to true and store it as a view. Join this view with the blocked table to get the administrator email of the one who blocked them. Project the user information and administrator information for the administrator who wanted to view the blocked users.
- Example = select user_email where is_blocked = ‘true’ from users;
    - select admin_email, user_email, user_username where user_email = (IN) blocked_emails from block table
    - Select block information from user where block is true

Delete Report:

- Data = administrator selects a report number to delete
- Requirements = report number must exist in report table and the report should be deleted from both the report and specific report type tables
- Select the report type for the chosen report number and get the corresponding specific report record (from either metadata_edit_request or transcript_error table) and delete the specific report. Then delete the tuple in the report table. To maintain referential integrity, cascade will have to be used to change invalid foreign keys after the delete to null.
- Example = delete report #123 from metadata_edit_request;
    - delete report #123 from report table
    - Delete report #123 from report table

Design a complete set of SQL queries to satisfy the transaction requirements identified in the previous stages, using the relational schema and views defined in tasks 2 and 3 above.
SELECT

Keyword Search

- SELECT audio_file_id
- FROM KEYWORDS
- WHERE search = keyword;

- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE
- WHERE audio_file_id IN files;

Interviewer Search

- SELECT interviewer_id
- FROM INTERVIEWER
- WHERE search_name = name;

- SELECT audio_file_id

- FROM INTERVIEW_BY
- WHERE interviewer_id IN interviewers;
-
- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE
- WHERE audio_file_id IN files;

Interviewee Search

- SELECT interviewee_id
- FROM INTERVIEWEE
- WHERE search_name = name;

- SELECT audio_file_id
- FROM INTERVIEW_OF
- WHERE interviewee_id IN interviewees;

- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE
- WHERE audio_file_id IN files;

Race Search

- SELECT interviewee_id
- FROM INTERVIEWEE_RACE
- WHERE searched_race = race;

- SELECT audio_file_id
- FROM INTERVIEW_OF
- WHERE interviewee_id IN interviewees;

- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE
- WHERE audio_file_id IN files;

City/State Search

- SELECT interviewee_id
- FROM INTERVIEWEE
- WHERE searched_city = city AND searched_state = state;

- SELECT audio_file_id
- FROM INTERVIEW_OF

- WHERE interviewee_id IN interviewees;

- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE
- WHERE audio_file_id IN files;

Report View (Searching for Metadata Reports)
- SELECT report_num
- FROM REPORT
- WHERE searched_type = type;

- SELECT report_num, user_email, description, decision
- FROM (reports NATURAL JOIN METADATA_EDIT_REQUEST);

Block View
- SELECT user_email, user_username
- FROM (USER NATURAL JOIN BLOCK);

- SELECT user_email, user_username, a_email, a_username
- WHERE user_email IN blocked_users;

Audio_File View
- SELECT title, date_published, audio_file_id, description
- FROM AUDIO_FILE;

INSERT
- INSERT INTO AUDIO_FILE
- VALUES (date_published, publisher, a_language, date_interviewed, title, description, file);

- INSERT INTO INTERVIEWER
- VALUES (gender, age, name);

- INSERT INTO INTERVIEWEE
- VALUES (gender, age, name, city, state);

- INSERT INTO REPORT
- VALUES (Audio_ID, type, user_email, a_email, decision, date_created);

- INSERT INTO USER

- VALUES (user_email, user_username, user_password, role);

- INSERT INTO CASUAL_USER
- VALUES (user_email, is_blocked);

- INSERT INTO ADMIN
- VALUES (a_email);

- INSERT INTO S_ADMIN
- VALUES (sa_email);

- INSERT INTO TRANSCRIPT
- VALUES (audio_file_id, transcriber, t_language, file);

- INSERT INTO KEYWORDS
- VALUES (audio_file_id, keyword);

- INSERT INTO INTERVIEWER_RACES
- VALUES (interviewer_id, race);

- INSERT INTO INTERVIEWEE_RACES
- VALUES (interviewee_id, race);

- INSERT INTO TRANSCRIPT_ERROR
- VALUES (report_num, error_description);

- INSERT INTO METADATA_EDIT_REQUEST
- VALUES (report_num, metadata_edit_description, metadata_name);

DELETE

- DELETE FROM AUDIO_FILE
- WHERE audio_file_id = id_to_del;

- DELETE FROM INTERVIEWER
- WHERE interviewer_id = int_id_to_del;

- DELETE FROM INTERVIEWEE
- WHERE interviewee_id = int_id_to_del;

- DELETE FROM REPORT

- WHERE report_num = rnum_to_del;

- DELETE FROM USER
- WHERE user_email = u_email_to_del;

- DELETE FROM TRANSCRIPT
- WHERE audio_file_id = id_to_del;

- DELETE FROM KEYWORDS
- WHERE audio_file_id = id_to_del;

- DELETE FROM INTERVIEWER_RACES
- WHERE interviewer_id = int_id_to_del;

- DELETE FROM INTERVIEWEE_RACES
- WHERE interviewee_id = int_id_to_del;

- DELETE FROM TRANSCRIPT_ERROR
- WHERE report_num = rnum_to_del;

- DELETE FROM METADATA_EDIT_REQUEST
- WHERE report_num = rnum_to_del;

UPDATE

- UPDATE AUDIO_FILE
- SET (title = new_title, description = new_descrip)
- WHERE audio_file_id = desired_id;

- UPDATE INTERVIEWER
- SET (name = new_name, gender = new_gender)
- WHERE interviewer_id = desired_id;

- UPDATE INTERVIEWEE
- SET (name = new_name, gender = new_gender)
- WHERE interviewee_id = desired_id;

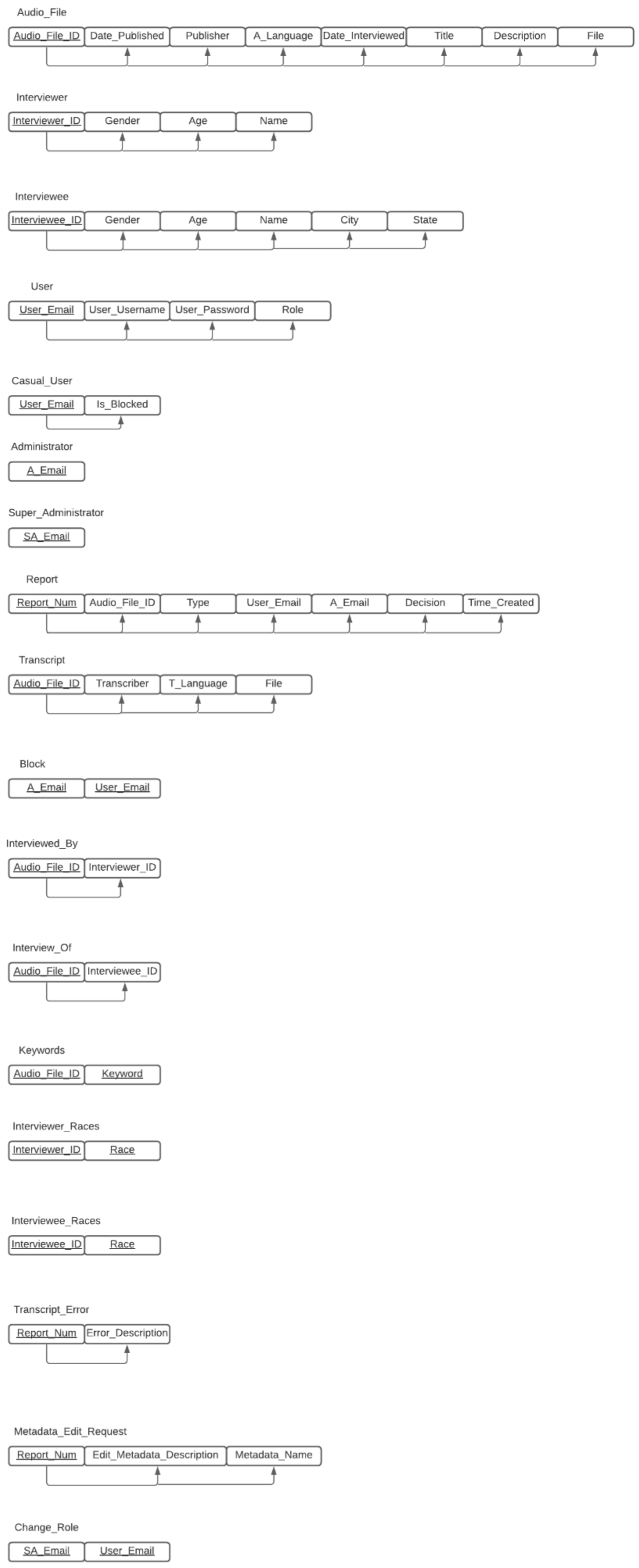
- UPDATE USER
- SET (user_username = new_username, user_password = new_password)
- WHERE user_email = desired_email;

- UPDATE TRANSCRIPT
- SET (transcriber = new_transcriber, t_language = new_language)
- WHERE audio_file_id = desired_id;

- UPDATE KEYWORDS
- SET (keyword = new_keyword)
- WHERE audio_file_id = desired_id;

- UPDATE INTERVIEWER_RACES
- SET (race = new_race)
- WHERE interviewer_id = desired_id;

- UPDATE INTERVIEWEE_RACES
- SET (race = new_race)
- WHERE interviewee_id = desired_id;


Audio_File
Audio_File_ID | Date_Published | Publisher | A_Language | Date_Interviewed | Title | Description | File
Interviewer
Interviewer_ID | Gender | Age | Name
Interviewee
Interviewee_ID | Gender | Age | Name | City | State
User
User_Email | User_Username | User_Password | Role
Casual_User
User_Email | Is_Blocked
Administrator
A_Email
Super_Administrator
SA_Email
Report
Report_Num | Audio_File_ID | Type | User_Email | A_Email | Decision | Time_Created
Transcript
Audio_File_ID | Transcriber | T_Language | File
Block
A_Email | User_Email
Interviewed_By
Audio_File_ID | Interviewer_ID
Interview_Of
Audio_File_ID | Interviewee_ID
Keywords
Audio_File_ID | Keyword
Interviewer_Races
Interviewer_ID | Race
Interviewee_Races
Interviewee_ID | Race
Transcript_Error
Report_Num | Error_Description
Metadata_Edit_Request
Report_Num | Edit_Metadata_Description | Metadata_Name
Change_Role
SA_Email | User_Email

Construction: **Tables, Queries, and User Interface:**

- Tables and Queries: https://github.com/Gabscurioshop/trentaudio/tree/main/src/setup

- Sample Display of Tables**:** https://github.com/Gabscurioshop/trentaudio/wiki/Tables-and-Queries

- User Interface: https://github.com/Gabscurioshop/trentaudio/tree/main/src/app

- Sample Display of UI: https://github.com/Gabscurioshop/trentaudio/wiki/User-Interface

Transition: Maintenance
Link to source code: https://github.com/Gabscurioshop/trentaudio/tree/main/src

Transition:
Link to public repo: https://github.com/Gabscurioshop/trentaudio