Group 6
Trentoniana Timeline

Jack Liddy, Christopher Smith, Eain Kelton, Aniello Sabatino

Emails: liddyj1@tcnj.edu, smithc69@tcnj.edu, keltone1@tcnj.edu, sabatia2@tcnj.edu

Inception: "Executive Summary"

This will be about one or two pages long. It should include:

- Need: the important stakeholders and market need your group identified
    - Our group was tasked with creating a comprehensive database containing various different file types from the Trentoniana Project. The main need was chronologically ordering the audio files presented on the Trentoniana Project. Users should be provided with a visual representation of these files in an appealing format. The stakeholders of this project are Trentoniana itself, our group members, the TCNJ Computer Science Department, and the users of the database.
- Approach: your unique and defensible approach
    - Our group's solution to this problem was to create a visual, interactive timeline. Our interface would be easy to use and access. Also, it would require little to no prior knowledge on the user's part. This way, users can both easily find what information they are looking for, while learning the context of the information, and surrounding events in the process.
- Benefits: the value of your product when compared to the status quo or alternatives
    - The benefits of our group's solution are as follows. The first benefit would be that our solution creates comprehensive and organized data for users. This is because our timeline provides a chronological list of audio files for users. Another benefit would include that users have the opportunity to see other relevant information that occurred to the data they are looking for. Some examples of relevant information the user can see would include the audio file name, size, and date when the file was created. A final benefit would be that our interface would be easy to use and navigate. In our implementation, we have a help button which the user can click on and it will give information on where each link will take the user, and instructions on how to use the website. Users can easily access audio files by clicking on the play button and if they wish to navigate to the page of the audio file a url is provided as well.
- Cost: the stakeholder cost to implement, e.g. would your approach replace an existing website, be an extension to an existing website, or be a separate new website?
    - Our project can be added to the existing Trentoniana website. However, this integration would come at the cost of time and money for Trentoniana to implement it. This is because it would require some to implement our tables into the Trentoniana database. It would require a paid worker to join or create new tables to store our information. Also, there would be a low cost in storage since our database will not take up a large amount of space with the number of entries we provided. However, the addition of more entries and audio files would take up space in the Trentoniana database system. Also, since we created a python flask application, this would have to be implemented into the Trentoniana website. Our application can be implemented into the Trentoniana web framework by using Gunicorn. This python web gateway server is broadly compatible with many different web frameworks. It also is light on server resources, so this

implementation would have a low cost on the overall performance of the Trentoniana website.

Elaboration: Project Proposal and Specifications

This consists of the materials submitted for Stage II, with revisions clearly identified.

Project Repository: https://github.com/TCNJ-degoodj/stage-2b-group-6

Stage II – Project Proposal Pitch and Specifications

Problem Statement

The data on the Trentoniana database is not well organized, and is difficult for the user to sort through and interpret. Specifically, it is difficult to view the data in chronological order and get a sense of what data corresponds to a certain era. It is also difficult for the user to view multiple kinds of data from a given era at once.

Objective of the module

A solution for this problem would be to implement a timeline organizing the data chronologically. This will give the user a sense of time when analyzing or searching for data. By doing this, the user's understanding of the data will improve, particularly to provide a proper framing of the data. We will focus on the dates mentioned within the transcript to create the historical timeline.

Description of the desired end product

The database as an end product would have an interactive, clean, and intuitive interface which organizes data in a reasonable way. The data would include tags which allow them to be categorized in a way that is readily understandable to the user, including time-stamps which will provide a sense of chronology. The interactive timeline will provide specifically chronological organization, allowing the user to have a visual representation of this aspect of the data.

Description of the importance and need for the module

The fact that the data are not organized chronologically means that it is hard to find data from a specific time period, and those listed near each other could be from different time periods. This may lead to difficulty for a user attempting to find or analyze data, in part due to the different speech mannerisms apparent across different time periods. By sorting data chronologically, users will have an easier time understanding what time period to expect before accessing the data directly. The interactive timeline will also allow them to easily progress through the different time periods in either direction.

Plan

______We plan on downloading as much data as possible from the Trentoniana website, specifically audio files, noting the date referenced in each piece of data. We then plan on organizing the data and audio files into chronological order, and presenting a visual representation of the time period that each piece of data had occurred at. We plan to visualize the data in a timeline that will allow the user to put into perspective when each event took place, and where it stands relative to other pieces of data.

Other similar systems / approaches that exist

______A similar system that currently exists is the New York Public Library Community Oral History Project. This system organizes their audio files alphabetically by the name of the interviewee. This makes it difficult to grasp how these events fit chronologically since they are alphabetized rather than organizing by date. Our approach will give the user a better sense of how the dates these stories fit in history and will be able to compare data with similar dates. It will also provide a visual representation to make it easier for the user to interpret the data chronologically.

Possible other applications of the system

______Our system could contain several different functionalities such as the ability to filter by different criteria. This would allow us to organize our database in such a way that we store data based on these criteria, and users can pick what information would be relevant to them.

Performance

The database would have to be efficient at retrieving and searching for data. The algorithms that will be created in order to retrieve audio files should be efficient so that there will not be any noticeable delays when playing audio files.The database should be able to handle multiple users pulling data at the same time. There shouldn't be any crashes when data is being pulled for different requests. The timeline should also initially load, displaying icons representing each available record of data, then upon selecting the record, the complete piece of data should display.

Security

We plan to implement minimum password lengths in order to make user and admin accounts more secure. We can also eliminate the use of single quotation characters in order to prevent any SQL injection attacks. Furthermore, we're planning to monitor user logins to make sure there aren't any attempts at code injection into our website. User accounts will have less permissions than admin accounts. For example, they will not be able to change any data on the website, users will only be able to view and search through different audio files on the timeline.

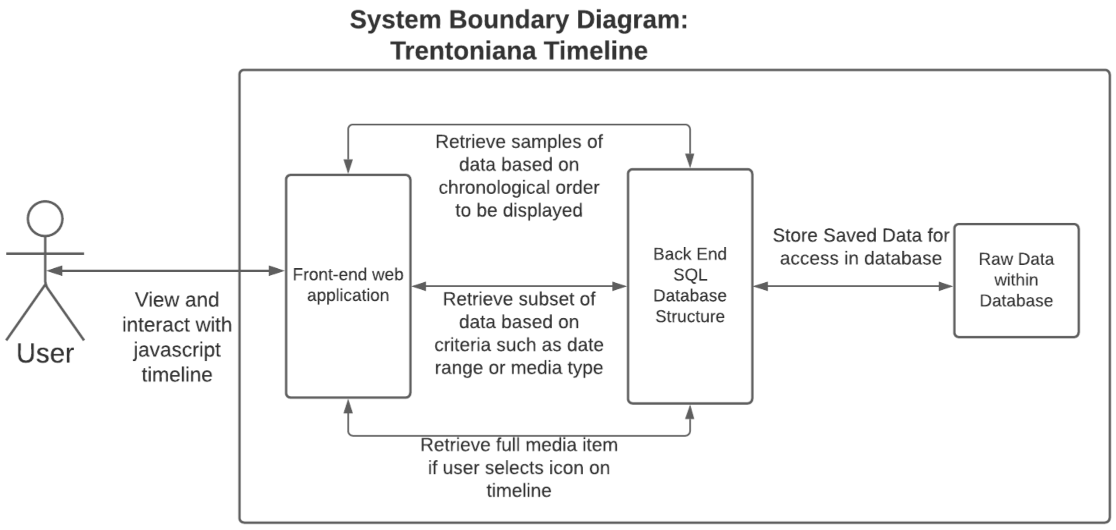
Backup and recovery

Will be able to use the basebackup tool for PostgreSQL to provide a complete set of backup files for our database. This will ensure that we will have a base backup for all the essential data that is stored on our website. We plan to have at least one base backup, we may create more if needed.

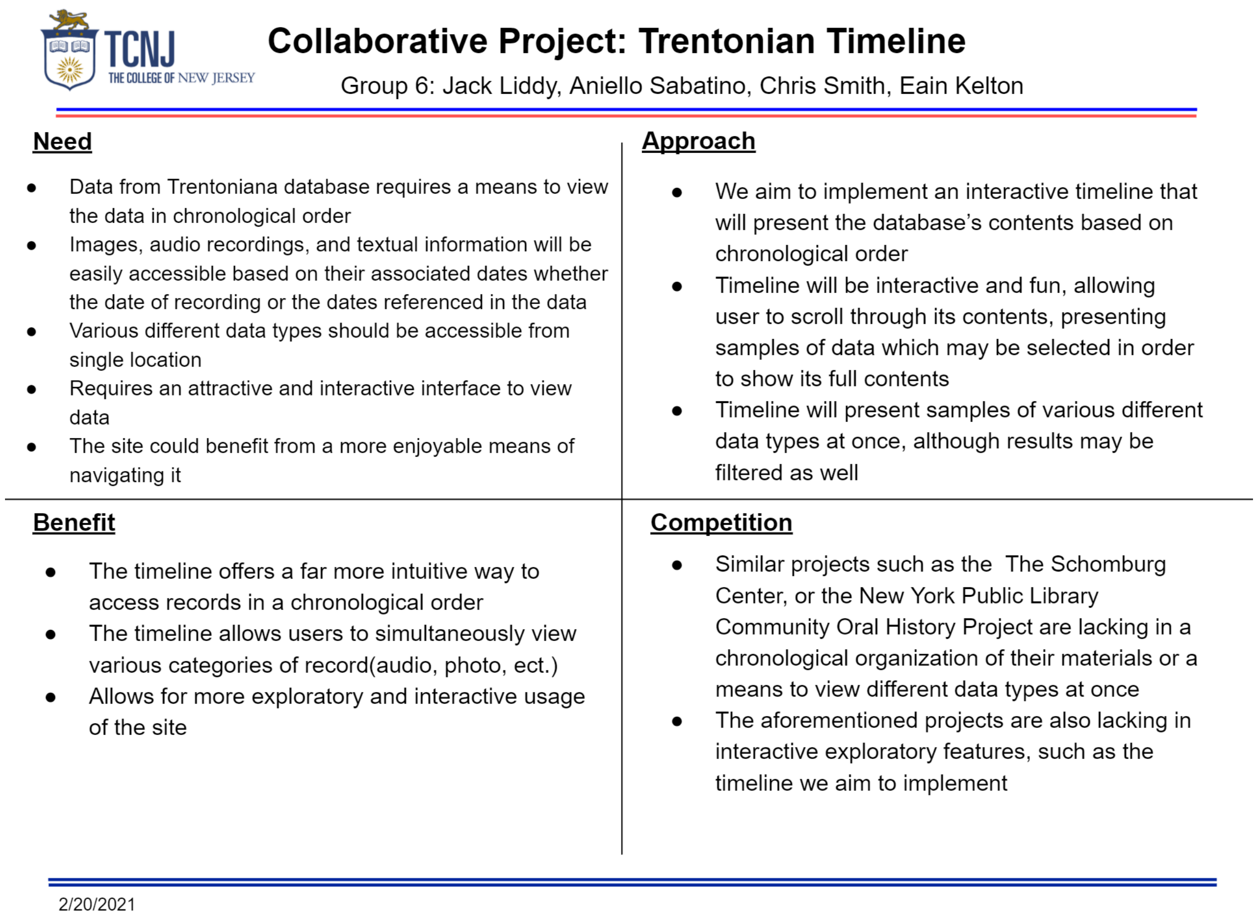
Technologies and database concepts the team will need to learn

______The team will need to learn how to properly create a database and store data in PostgreSQL. Also, team members will need to learn how to create a backup database in PostgreSQL. The database will ultimately be accessed through a web application, so the team must incorporate some sort of web interface to communicate with the database as well as implement our desired functionality. When implementing the timeline, we plan on using a javascript library, such as d3.js, which should simplify the creation of an interactive timeline based on our database's contents.

A diagrammatic representation of the system boundary


System Boundary Diagram: Trentoniana Timeline
User
View and interact with javascript timeline
Front-end web application
Retrieve samples of data based on chronological order to be displayed
Retrieve subset of data based on criteria such as date range or media type
Retrieve full media item if user selects icon on timeline
Back End SQL Database Structure
Store Saved Data for access in database
Raw Data within Database

Quad Chart



Collaborative Project: Trentonian Timeline

Group 6: Jack Liddy, Aniello Sabatino, Chris Smith, Eain Kelton

Need

- Data from Trentoniana database requires a means to view the data in chronological order
- Images, audio recordings, and textual information will be easily accessible based on their associated dates whether the date of recording or the dates referenced in the data
- Various different data types should be accessible from single location
- Requires an attractive and interactive interface to view data
- The site could benefit from a more enjoyable means of navigating it

Approach

- We aim to implement an interactive timeline that will present the database's contents based on chronological order
- Timeline will be interactive and fun, allowing user to scroll through its contents, presenting samples of data which may be selected in order to show its full contents
- Timeline will present samples of various different data types at once, although results may be filtered as well

Benefit

- The timeline offers a far more intuitive way to access records in a chronological order
- The timeline allows users to simultaneously view various categories of record(audio, photo, ect.)
- Allows for more exploratory and interactive usage of the site

Competition

- Similar projects such as the The Schomburg Center, or the New York Public Library Community Oral History Project are lacking in a chronological organization of their materials or a means to view different data types at once
- The aforementioned projects are also lacking in interactive exploratory features, such as the timeline we aim to implement

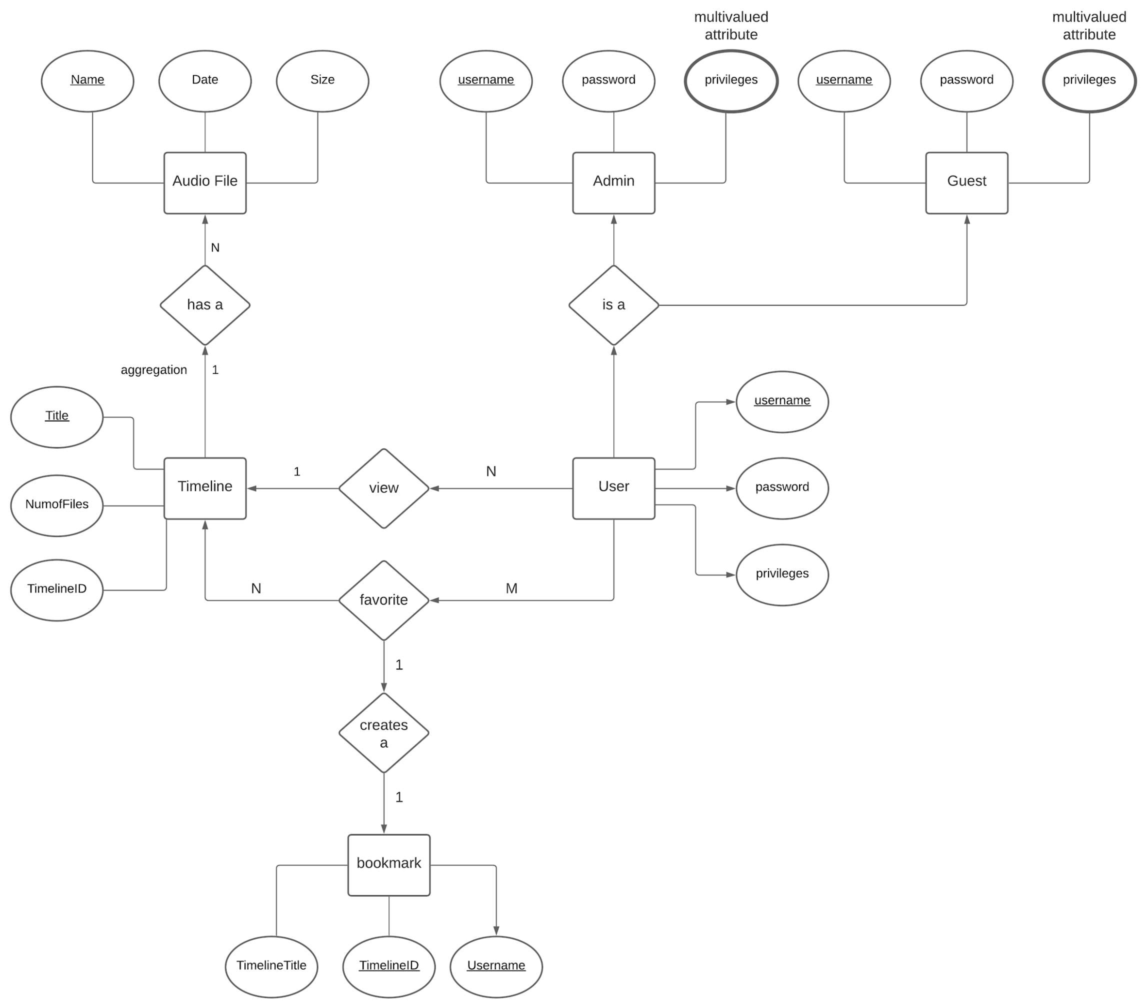
2/20/2021

Elaboration: Design

This consists of the materials submitted for StagesIII and IV, with revisions clearly identified.

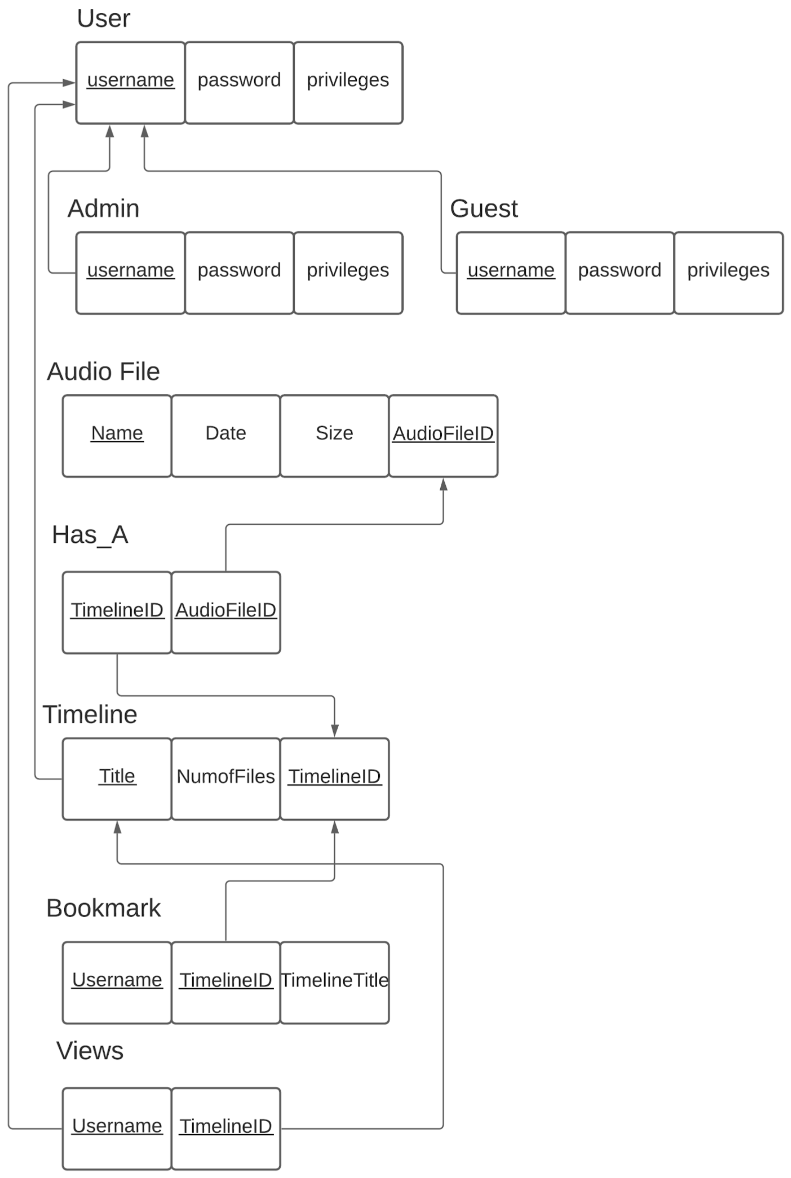
Stage III: Database Model

ER Diagram



Relational Schema

Relational Schema



Database Size and Search Estimates:

- There are 74 audio records included in the Trentoniana Project's archive. Considering possible additions over time, this can reasonably be estimated as about 100 records

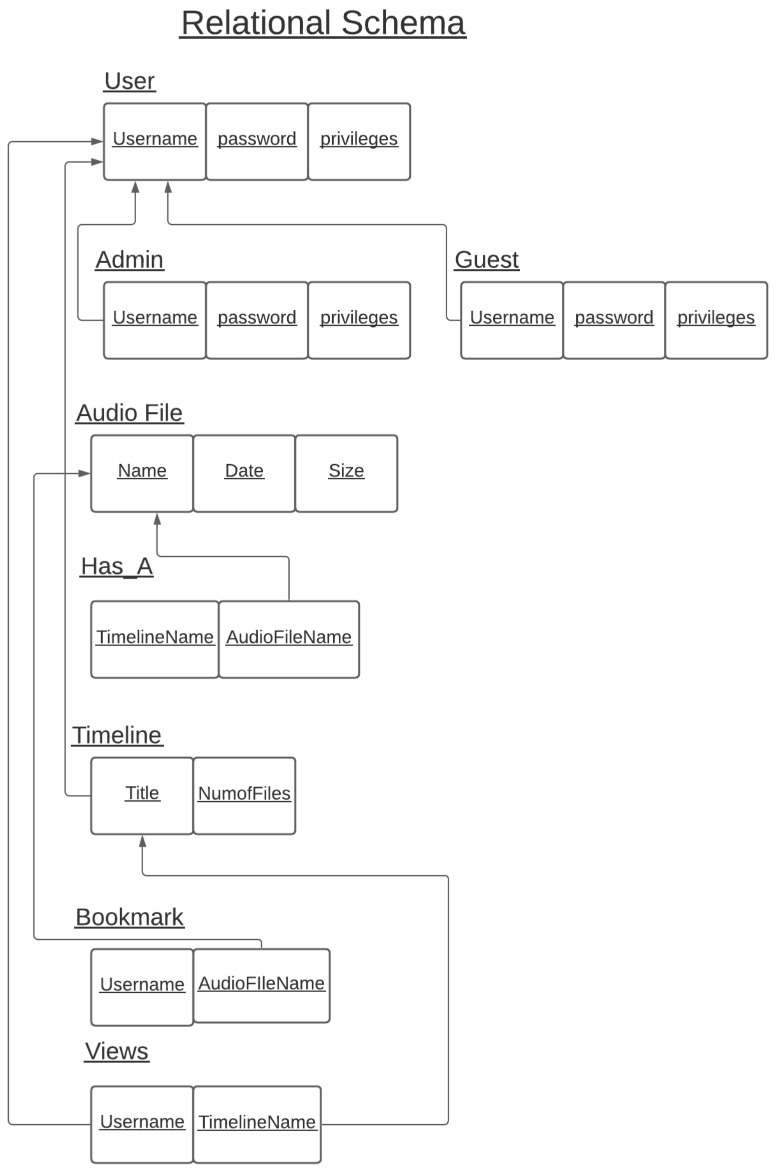
- The records can be filtered 13 ways by year, 5 ways by topic & subject, 2 ways by collection, 1 way by media type and can be searched by title. This totals 22 different ways of filtering/searching the records. Many topic & subject items currently included as filters are redundant, and could be removed, making 3 a reasonable estimate. Records may be added in the future with years not currently included, with 20-30 year options being a good estimate (including undated records). One collection option is labeled "favorites" and does not seem necessary. Considering the Trentoniana Project's archive is considered a collection, only 1 collection option is needed. Since the only medium of these records is audio, this means only 1 media option is needed, and of course searching by title will remain 1 option. Therefore, about 25-35 filtering/searching types would be needed.

- Given that the project is relatively small-scale, with only 74 records and a rather small audience, it's reasonable to estimate the database will have about 100 searches per day.

<u>Stage IV – Elaboration: Design</u>

1. Review the Database Model document with stakeholders, and update the model as needed.

This is our updated relational schema, we deleted the attributes that we removed in order to normalize our tables to BCNF.



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

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

User

Username	**password**	**privileges**

chrissmith42	lion3	admin
jackliddy5	windows5	admin
guest46	linux3	guest
newuser55	mac05	guest

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: No partial dependencies: Username is the only part of the candidate key**
- **3NF: No transitive dependency: there are no indirect relationships causing a functional dependency**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

Audio File

Name	**Date**	**Size**	**AudioFileID**
JHS 49	2016-10-13	37.9 MB	A1
JHS 33	2016-10-15	43.4 MB	A2
Segal, Mark Z.	1998-06-01	33.6 MB	A3
Cohen, Chester	1998-06-18	43.4 MB	A4

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: There is a partial dependency since Date and Size can be derived by just AudioFileID and it does not need to refer to both Name and AudioFileID.**

Normalizing to BCNF

Name	**Date**	**Size**
JHS 49	2016-10-13	37.9 MB
JHS 33	2016-10-15	43.4 MB
Segal, Mark Z.	1998-06-01	33.6 MB
Cohen, Chester	1998-06-18	43.4 MB

- **3NF: No transitive dependency: there are no indirect relationships causing a functional dependency**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

Timeline

Title	**NumOfFiles**	**TimelineID**
Oral History of Trenton	75	T1
Jewish History	65	T2

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: Partial dependencies: There is a partial dependency since NumOfFiles can be derived by just TimelineID it does not need to refer to both Title and TimelineID.**
- **This table is not in the BCNF because it has partial dependencies.**

Normalizing to BCNF

Title	**NumOfFiles**
Oral History of Trenton	75
Jewish History	65

- **We decided to normalize the table to BCNF by removing the TimelineID column since it is unnecessary to have if we do not allow duplicates for the Title attribute.**
- **3NF: No transitive dependency: there are less than three columns so it is not possible to have transitive dependencies.**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

Has_A

TimelineName	**AudioFileName**
T1	{A1,A2,A3,A5,A7}
T2	{A5,A7,A9,A11,A14}

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: No partial dependencies: TimelineID is the only part of the candidate key**
- **3NF: No transitive dependency: there are less than three columns**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

Bookmark

Username	**TimelineID**	**TimelineTitle**
chrissmith42	T1	Oral History of Trenton
jackliddy5	T2	Jewish History
guest46	T2	Jewish History
newuser55	T1	Oral History of Trenton

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: Partial dependencies: There is a partial dependency since TimelineTitle can be derived by just using TimelineName**
- **This table is not in the BCNF because it has partial dependencies.**

Normalizing to BCNF

Username	**AudioFileName**
chrissmith42	{A1,A2,A3,A5,A7}
jackliddy5	{A5,A7,A9,A11,A14}
guest46	{A5,A7,A9,A11,A14}
newuser55	{A1,A2,A3,A5,A7}

- **We decided to normalize the table to BCNF by removing the TimelineID and TimelineTitle columns since we decided to bookmark audio files instead. Users will only bookmark audio files, not timelines in this new approach.**
- **3NF: No transitive dependency: there are less than three columns so it is not possible to have transitive dependencies.**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

Views

Username	**TimelineName**
chrissmith42	Oral History of Trenton
jackliddy5	Jewish History
guest46	Jewish History
newuser55	Oral History of Trenton

Is it in BCNF?

- **1NF: All values are atomic, column names unique, values in each column belong to same domain**
- **2NF: No partial dependencies**
- **3NF: No transitive dependency: there are less than three columns**
- **BCNF: Passes all other forms and all non prime attributes have a superkey**

3. 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.

Order audio files chronologically in the Audio File table

chronological_view

Name	**Date**	**Size**
Segal, Mark Z.	1998-06-01	33.6 MB
Cohen, CHester	1998-06-18	43.4 MB
JHS 49	2016-10-13	37.9 MB
JHS 33	2016-10-15	43.4 MB

Select all usernames that are admins in the User table

admin_view

Username	**password**	**privileges**

chrissmith42	lion3	admin
jackliddy5	windows5	admin

Select all usernames that are guests in the User table

guest_view

Username	**password**	**privileges**
guest46	linux3	guest
newuser55	mac05	guest

Display all bookmarks for a specific username(ex: chrissmith42)

User_bookmarks_view

Username	**TimelineName**
chrissmith42	Oral History of Trenton

4. 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.

chronological_view

```
CREATE VIEW chronological_view AS
SELECT * FROM Audio File
ORDER BY Date
```

admin_view

```
CREATE VIEW admin_view AS
SELECT * FROM User
WHERE privileges=admin
```

guest_view

```
CREATE VIEW guest_view AS
SELECT * FROM User
WHERE privileges=guest
```

User_bookmarks_view

```
CREATE VIEW User_bookmarks_view AS
SELECT * FROM Bookmark
WHERE Username=chrissmith42
```

Construction: Tables, Queries,and User Interface

This consists of the submissions for Stage V, with revisions clearly identified.

Stage Va – Construction

1.Write and execute SQL data definition language(DDL)commands to create the tables and views in PostgreSQL. Ensure that all constraints are specified.

```
createdb TrentonianaTimeline
psql TrentonianaTimeline
CREATE TABLE users (username char(30) PRIMARY KEY,password text,privileges text);
CREATE TABLE AudioFiles (name char(30) PRIMARY KEY,date DATE,size text);
CREATE TABLE Timeline (name char(30) PRIMARY KEY,numOfFiles text);
CREATE TABLE Has_A (TimelineName char(30) PRIMARY KEY,AudioFileName text);
CREATE TABLE Bookmark (username char(30) PRIMARY KEY,audiofileName text);
CREATE TABLE Views (username char(30) PRIMARY KEY,TimelineName text);
```

2.Write scripts / programs that may be required to obtain and format the data.

3.Populate the tables with valid data, with all constraints being enforced.

```
INSERT INTO users
VALUES ('chrissmith42','lion3','admin'), ('jackliddy5','windows5','admin'),
('guest46','linux3','guest');

INSERT INTO AudioFiles
VALUES ('JHS 49','2016-10-13','37.9 MB'), ('JHS 33','2016-10-15','43.4 MB'), ('Segal, Mark
Z.','1998-06-01','33.6 MB');

INSERT INTO Timeline
VALUES ('History of Trenton','75'), ('Jewish History','65'), ('Oral History', '66');
```

```
INSERT INTO Has_A
VALUES ('History of Trenton','JHS 49'), ('Oral History','JHS 59'), ('Jewish History','JHS 59');

INSERT INTO Bookmark
VALUES ('chrissmith42','JHS 49'), ('jackliddy5','JHS 59'), ('guest46','JHS 33');

INSERT INTO Views
VALUES ('chrissmith42','History of Trenton'), ('jackliddy5','Jewish History'), ('guest46','Jewish
History');
```

4.Write SQL data manipulation language(DML)commands that were designed in the previous stages. The queries must be elegant and make effective use of complex query constructs such as subqueries. Execute and test these queries to ensure that they work correctly. Examine the outputs carefully to verify that the queries do not return spurious tuples.

chronological_view

```
CREATE VIEW chronological_view AS
SELECT * FROM AudioFiles
ORDER BY Date
```

admin_view

```
CREATE VIEW admin_view AS
SELECT * FROM user
WHERE privileges='admin'
```

guest_view

```
CREATE VIEW guest_view AS
SELECT * FROM users
WHERE privileges='guest'
```

user_bookmarks_view

```
CREATE VIEW user_bookmarks_view AS
SELECT * FROM Bookmark
WHERE username='chrissmith42'
```

5.Create issues to assign tasks to each team member and update existing ones. Create new issues for tasks that need to be completed or can be added in a future version of the application.

FULL SCRIPT

```
createdb TrentonianaTimeline
psql TrentonianaTimeline
CREATE TABLE users (username char(30) PRIMARY KEY,password text,privileges text);
INSERT INTO users
VALUES ('chrissmith42','lion3','admin'), ('jackliddy5','windows5','admin'),
('guest46','linux3','guest');

CREATE TABLE AudioFiles (name char(30) PRIMARY KEY,date DATE,size text);
INSERT INTO AudioFiles
VALUES ('JHS 49','2016-10-13','37.9 MB'), ('JHS 33','2016-10-15','43.4 MB'), ('Segal, Mark
Z.','1998-06-01','33.6 MB');

CREATE TABLE Timeline (name char(30) PRIMARY KEY,numOfFiles text);
INSERT INTO Timeline
VALUES ('History of Trenton','75'), ('Jewish History','65'), ('Oral History', '66');

CREATE TABLE Has_A (TimelineName char(30) PRIMARY KEY,AudioFileName text);
INSERT INTO Has_A
VALUES ('History of Trenton','JHS 49'), ('Oral History','JHS 59'), ('Jewish History','JHS 59');

CREATE TABLE Bookmark (username char(30) PRIMARY KEY,audiofileName text);
INSERT INTO Bookmark
VALUES ('chrissmith42','JHS 49'), ('jackliddy5','JHS 59'), ('guest46','JHS 33');

CREATE TABLE Views (username char(30) PRIMARY KEY,TimelineName text);
INSERT INTO Views
VALUES ('chrissmith42','History of Trenton'), ('jackliddy5','Jewish History'), ('guest46','Jewish
History');


SELECT * FROM Views WHERE username IN (SELECT username FROM Views WHERE
TimelineName=’Jewish History’)
```

Transition: Maintenance

This is satisfied if the source code is appropriately documented and uploaded on GitHub, and the final report is appropriately labeled and organized.