**Wentworth Institute of Technology**

College of Engineering and Technology  
Department of Computer Science and Networking

Spring, 2018: Comp 2650, Databases

Dave Rosenberg, Proprietor, Dave Rosenberg Company

**Group Project – Final Write-up  
Group 13 – Media Center Application**

**Executive Summary:**

**[This summary should be several paragraphs, elaborating on the summary from the Group Proposal.  Remember, you’re building solutions to business problems (real or imagined). Your solution is an application (suite) which utilizes a database to store data/information to support your business.  From a business perspective, if you are trying to sell your idea to a business, this is the part your potential audience will read – the rest is only interesting if this section grabs them and draws them in.]**

This application is a media platform which allows for the sharing and monetization of media. Creators can add their media to the application for users to view. This enables distribution of media such as music from artists, videos from production companies, or images from professional photographers. Monetization is on a per-view basis with a changeable value.

This application utilizes a file server to keep the database easy to maintain. Any transfers are handled solely by the file server, and the only information contained by the database is file names. This keeps the application safe from constant downloads.

This application allows for users to request creator permissions. Allowing users to become contributors to the platform can allow for excellent financial opportunities as some creators become marketable, not unlike YouTube.

    An Admin is registered in the database. This admin has total control over what users become creators, what media is on the platform (with the ability to delete or add new media), and how much everyone is paid. Based on the current rate of pay, the admin can see the total payout for each creator.

    The creators of the platform content can see how much they are being paid, as well as statistics and finances for themselves. Adding music, videos, or images is an easy task with this application, making lives easier for the creators, encouraging future submissions.

    Overall, this application allows for a thriving platform with many different creators and users who allow for great financial opportunities, including user service fees and eventual creator marketability. Being easily modifiable allows this application to adjust to a ever-changing market and business.

**User Categories:**

**[Elaborate on the 2 or 3 categories of user from the Group Proposal.  Create a subsection for each user category, naming the category and describing the functions they perform within the scope of your application.  Create a sub-subsection for each function describing the information required and the types of access required for each.]**

User:

Views all created media (Requires select access to Media table)

Adjusts account settings (Requires update access to User table)

Creator:

    Functions of User category

    Adds/Deletes created media (Requires update access to Media table)

    Views finances (Requires select access to Creator table)

Admin:

    Functions of User category

    Functions of Creator category

    Adds/Deletes all media (Requires update access to Media table)

    View overall finances (Requires select access to Creator table & Media table)

    Accept users creator status request (Requires update access to Creator table & User table)

**Application Design Process:**

**[In a few paragraphs, describe your journey from concept to proposal to design to implementation.  What features/functions did you cut to get the project to a manageable size? What tradeoffs did you make to go from your ‘grand design’ to your representative implementation?]**

In the beginning of our proposal, we had functions we were planning to add but ultimately cut them in the end. A following function that allowed users to follow creators and their content. A favorite feature that allowed users to specifically favorite a piece of media and by doing that, the users would receive similar media. We couldn’t include these functions mainly due to technical errors.

Technical errors on Github would not allow us to push our code properly. This caused everyone to delete and re-import projects every time any of us needed the most recent pushed code. Also, code could only be tested locally, which was one person in our group, so we had to keep sending code back and forth to test.

This led to time being an issue as well. The favorite and follow function had to be cut. We didn’t substitute those functions for anything else, as the finished product right now was worked on till the end.

**Database Design Process:**

**[In a few paragraphs, describe the evolution of your database design from initial concept and Enterprise Diagram to EERD to normalized schema.  What did you need to add/remove to make the scope of the project doable in the allotted time? What did you need to tweak in the database as you implemented your application code?**

**What are the most useful SQL features for your application?]**

Initially, the database had a lot of metadata information relating to media, as well as the BLOBs of media files. After some review of the importance of this metadata (which ranged from runtime to dimensions), such information didn’t seem needed to make the database function correctly. A lot of time was spent making the database smaller and more manageable.

    The database went through many changes mid-development, due to sudden design flaws creeping their way during production. These include making username unique and eliminating the need for Media ID all together by making the title of the media the primary key. BLOBs were attempted, although it seemed difficult with the JDBC API. Thus, a file server was required for making large transfers possible. The database began accepting file names rather than binary data.

    Due to time constraints, a lot of the features currently in the database were never used. “TaggedMedia” was made to allow media be searchable by string tags, such as “horror” or “nature”. These cuts were to keep the project simple to ensure it could be done on time.

    Another victim of these cuts was any of the fancy database design pieces, such as views, triggers, and transactions. The database contains only tables with different keys. Time was too short for learning how to implement each specific feature, and to make it work well.

**About Your Data:**

**[What sources did you use to populate your database?  If you fabricated it, that’s your source. Be precise about the source (e.g. include complete URLs and the names of websites, not just the names or URLs of the home pages).  If you used data that I provided, consider me the source (for this project).**

**Describe the steps you took to manipulate the source format to be able to load it into your database.]**

*File “desert.mp3” (“Desert Song”): Newgrounds (*[*https://www.newgrounds.com/audio/listen/795431*](https://www.newgrounds.com/audio/listen/795431)*)*

This source was directly downloaded from the website using the supplied download button. That downloaded an MP3. The database only required the file name, “desert.mp3”. The binary data was given to the file server.

*File “rosenberg.png” (“The Truth”): Fabricated By Conner Theberge*

This source was created with Windows’ Paint application. The database only required the file name, “rosenberg.png”. The binary data was given to the file server.

*File “pirate.mp4” (“The Most Successful Pirate”): TED-Ed YouTube Channel (https://www.youtube.com/watch?v=6BALmDghybk)*

This source was downloaded from YouTube using a third-party downloader. The database only required the file name, “desert.mp3”. The binary data was given to the file server.

*User Information (Admin, creator)*

    Basic user information was fabricated and put into the database from the Media Center Application.

**Additional Analysis:**

**[How did you determine the integrity constraints for your model (domain/range of values, referential)?  How did you enforce them? Did you use Triggers and/or Stored Procedures to enforce the integrity constraints or to perform other tasks in a modular fashion?  How might you use Triggers/Stored Procedures if you were to continue development of your application?**

**How did you divide the project responsibilities amongst your group members?  How did you assist each other? What other resources did you use?]**

When we were thinking about how we wanted to implement and enforce integrity constraints on our database project we thought heavily about how the database would be used and what we needed to implement in order to protect it. We wanted the user to have the easiest possible time signing up and deleting their accounts so we made sure that we didn’t constrain them in that manner this was allowed by the user not having any dependant data so the existence of their account at any given moment didn’t really matter in the grand scheme. For the creator the they are also allowed to delete their accounts at anytime, like their user counterparts. Unlike the user group, the creator group has data that is dependant on their specific ID so special measures needed to be put into place in order to keep data without a creator out of the database. Due to hypothetical copyright constraints (if a creator doesn’t want to keep their account it is assumed they would like their IP’s to be removed as well) all of a creators data is removed from the database after its creator withdraws from the program. Lastly the Admin group which can act also as a creator is the only user type that cannot delete their account due to the obvious fact that they need to administrate the database and creator program in order for the the whole operation to work. We did not use any SQL triggers or stored procedures due to time constraints, we had already developed workarounds to these tools before they were introduced fully in class and did not have time to remove our workarounds and replace them. If we were to have more time we could have had features that would make it so that files could be (with the creators consent) kept on the database but automatically moved to a generic or creative common artist as a placeholder for lookups.

When we grouped up and formed Group 13 we didn’t anticipate the issues regarding the installation of the SQL tools and environments. Regardless of our best efforts only one of our groupmates, Conner, was able to get the SQL tools working. This forced us into an interesting situation where we had no way for us each to test our works and it ended up putting a bottleneck on the development process. Without anyone else being able to test the SQL it set in stone whom had to work on the SQL to Java connections. This left the rest of the interfaces and implementation of the SQL to Java connections that Conner worked on to our other two members. The process that we when through assigning these tasks wasn’t anything special both of the remaining members chose 2 of the “Scene” classes that needed to be done and the odd one out was given to Conner.

During our development cycle we tried to use GitHub to host our project and streamline our workflow. That was a short lived dream as it soon became apparent that there were some flaws how we wanted to use GitHub. This resulted in us having to send all of our updated classes to a central computer (Conner’s due to him being the only one with SQL working) via email for testing. This slowed us down but was all in all manageable. Because of how difficult it was for us to connect our projects up it resulted in us having to have Conner test and check that all of the features of our program were working as anticipated. Much of the bug squishing was a team/partnered effort with Conner because of how we had to rely on him to know how the other two members classes were playing with the database.

**Project Team**

Jason Moy: CreatorScene, MediaScene

Tyler Crosby: LoginScene, AdminScene

Conner Theberge: Query Classes, UserScene