# CS374 – Intro to Database Management

# Application Development Project

# Rubric for Status Update

## Group Member #1: Battista, Jude

## Group Member #2: Ouellette, Dean

## Group Member #3:

## Group Member #4:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Requirements | Points | Awarded |
| Description of Application | * An overview of your application * *changes since second deliverable* * System requirements (e.g. hardware, DBMS, other software) * A detailed description of your application * Are there features that will not be implemented? What are they, and why won’t you fulfill them? | 5 |  |
| Schedule | * *changes since second deliverable* * Detailed schedule of who will do what part of project, by when | 5 |  |
| Logical Diagram | * *changes since second deliverable* * Logical diagram in UML or E-R * Discussion of how your data model will satisfy the needs of your application * Discussion of alternative designs that you did not do (and why) | 5 |  |
| Queries Required | * *changes since second deliverable* * Required queries in English (not SQL) * What entities and/or relationships are required for each query? * How will each query satisfy the needs of your application | 5 |  |
| Grammar, punctuation, syntax, and references | * Follow rules from the Penguin handbook on writing * References as appropriate (e.g. if you are modeling your application after an existing application, make note of that) | 5 |  |

**Description of Application**

Application Overview: We intend to create an application enabling multiple users to easily access information about music, specifically focusing on jazz recordings. Users will be able to query this data using conditions such as artist, instrumentation, track or album titles, venue, and dates to find old favorites or discover new recordings of interest.

System Requirements:

Front end – In pursuit of our design goal of ease of access for multiple users, we intend to create a web-based front end created primarily in C# and Razor Pages .

Cloud Based – As we do not wish to administer our own database server, we will take advantage of Microsoft’s student access to Azure for our hosting needs.

C# server application – We wish to take advantage of Linq’s deterministic query features, so we will design the server side application in C#.

DBMS – SQL Server. We intend this to be compatible with a range of database systems, but development will focus on SQL server in order to take advantage of the Microsoft stack’s integration with Azure.

Application Description: The basic premise of the application is a database storing information about jazz recordings, more specifically information about albums, tracks on the albums, and artists, and the way these things relate. For artists, the important information is what albums they’re featured on (and whether they are a leader or a sideman on each album), which songs they perform (and whether they solo on these songs), and the instrument they play on each track. For albums, the important information is the length, the release date, and the tracks they contain. For tracks, the important information is the title, the duration, the date of recording, and the venue. The user will be able to query the system using one or more of these attributes as qualifiers and be shown a list of all artists, albums, or tracks fitting their specification(s). In addition, because there is no simple way to input every jazz album ever recorded into the database, users will also be able to add albums, and the associated artists, venues, tracks, and instruments.

Feature List:

Easy addition of albums

Customizable queries- combinations of:

Artists

Album titles

Track titles

Year

Venue

Changes since Deliverable 2 – On Microsoft’s advice, we moved away from an MVC front end using Javascript and JSON, and instead opted for Razor pages with underlying C# code. The collegiate Imagine subscription to Azure no longer permits users to host web applications or services, so we are relying on Dean’s laptop instead.

**Project Management Schedule**

20 Nov - Get sample MVC application running on Azure – Jude

20 Nov - First iteration of SQL server database running on Azure - Dean

1 Dec - Barebones UI view in JS and html - Both

4 Dec - Create configurable queries in LINQ on controller - Both

4 Dec - Add data transmission from UI - Both

8 Dec - Create script for adding data from CSV - Dean

8 Dec - Populate database with representative sample - Dean

8 Dec- Refine UI to a usable state – Jude

8 Dec - Make an attempt at securing the database from front end attacks – Jude

10 Dec - Testing - Both

14 Dec – Present - Both

**Logical Diagram**

..\ erdplus-diagram.png

We knew we needed to track information about albums, tracks, and authors as the foundations of a jazz library, so we created each of them as entities. After examining how each entity might interact with the others, we opted for an Artist-centric approach, as users are likely to prioritize an artist they enjoy over a specific track played in different ways by different artists. Thus Artists are featured on Albums, perform Tracks, and play Instruments. We considered an Album-centric approach, but Albums seemed more likely to be an output than a parameter: knowing an Album’s details means the user is likely to already possess information about the tracks and performers. Instead, it seems likely that a user will identify a track or an artist and search for albums featuring them.

We are still determining the best way to relate Instruments to Artists. For multi-instrument performers, it seems useful to allow the user to specify which instrument they are interested in. On the other hand, it could significantly impact our query result output.

**Queries Required**

From the user’s perspective we have three configurable queries. Internally, each will be constructed in LINQ via parameters fed in from the user.

Find track – Find tracks matching selected attributes to include artist name, track title, track date recorded, and the album on which the track was released. Return all matching tracks and display the associated track name, duration, and recording date.

Requires the following entities and relationships: Album, Featured On, Artist, Artist Performs On, Tracks, Albums Contain Tracks

This query is the primary focus of the project, enabling users to find tracks that match their parameters.

Once we’ve located a track, we offer supplemental queries to find all the albums that track was released on, as well as all the artists to have performed that track.

Find album – Find all albums matching selected attributes to include album name, an included track name, a featured artist, and the year the album was released.

This query is the secondary focus of the project, enabling the user to locate albums rather than individual tracks. We believe this will be most useful for users as they expand their collection, enabling them to do things like determine which albums to buy in order to access their desired tracks.

Once we’ve located the album, secondary queries allow the user to find all the artists featured on that album as well as all the tracks contained in the album.

Find artist - Find any artist with a matching first name or last name, who was featured on an album, or who performed a track.

Once we’ve located an artist, we also want to let the user find all tracks the artist has performed, as well as all albums the artist is featured on.

Changes since Deliverable 2: - We’ve moved away from allowing the user to input data. Quality control on user input data proved to be a task worthy of a project in and of itself. We still offer basic input options, but it’s no longer a focus of the project.

We’ve also split up our query structure to allow the user to focus on a single type of result (album, artist, or track) with secondary queries allowing the user to find records of the other two types associated with their primary result. We believe this offers a better user experience for the majority of the potential queries.