School of Computer Sciences
Matloob Khushi, Harshana Randeni

ISYS2120: Data and Information Management **Semester 2, 2020**

Assignment 03: DB Application Programming

1 Assessment Details

This assessment is worth 15% of your final grade.
This assessment is a group assessment.

2 Due Date

This assessment is due at **23:59:00** on **17th Nov 2020** .
Late assessments will be graded according to the late penalty guidelines outlined in section 6.1.

3 Group Work

This assignment is done in groups of up to 4 students (aim for exactly 4 members, but it may happen that sometimes a group is smaller or larger). We recommend that all students in a group be attending the same lab session, so you can work together more easily.
Procedure: In week 7 and 8 labs, you should form groups and pairs. Once you have your pairs, you can nominate another pair to pair with as long as the end result for the tutorial isn't more than two people per group having old team members.

What to do in the event of Group Issues: If, during the course of the assignment work, there is a dispute among group members that you can't resolve, or that will impact your group's capacity to complete the task well, you need to inform the unit coordinator, matloob.khushi@syney.edu.au. Make sure that your email names the group, and is explicit about the difficulty; also make sure this email is copied to all the members of the group. We need to know about problems in time to help fix them, so set early deadlines for group members, and deal with non-performance promptly (don't wait till a few days before the work is due, to complain that someone is not doing their share). If necessary, the coordinator will split a group, and leave anyone who doesn't participate effectively in a group by themselves.

4 Assessment

You will be assessed in three components for this assignment. The first two are submissions - a report and a code base. The final assessment is a demo.

4.1 Report

Produce a PDF, or docx file - please do not try to submit any other filetype as it will not work.
It should contain the following:

- Your SQL queries to solve the 10 DB tasks.
- Screen shots of your application of the pages produced by each functionality that you made
- Description and presentation of your added functionality including:
  - New SQL Functions in 'database.py'
  - New Routing Functions in 'routes.py'
  - New Templates in 'templates/*.html'
  - Modifications to any of the above or any other project files.
  - Screenshots of the functionality in action

This file should be submitted by only one member of the group, through the Canvas assignment link.

4.2 Code

A compressed file of your code base and a git link to your code hosted on the University GitHub Server. We will provide some materials on how to do this as well as some guidance during the week 9 tutorial after the quiz.

4.3 Demo

You will have a demo in your week 12 tutorial (Nov 18 - Nov 20). Attendance to this demo is **MANDATORY**. If you are unable to attend the demo (and unable to provide a case for misadventure), then you will be **scaled to a mark of 0 for this assessment**. If you know ahead of time that you may be unable to make it - please inform your tutor as soon as possible and we will arrange an alternate demo time.
The demo will assess group member participation. Each group member will be asked about different components of the assignment and you will be asked to demonstrate your new functionality. If it becomes clear that you are have not participated or done little of the work - we may scale your marks accordingly.

4.4 Assessment Mark Breakdown

The marking rubric will follow these guidelines (with some leeway if we judge the workload is too much on a particular area):

- **5 marks** If you complete the 10 SQL tasks and adequately report them

- **3 marks** If you complete the 4 routing and 4 rendering tasks and adequately report them
- **3 marks** If you add a significant new functionality (can't just be a copy of an existing one) and adequately report it.
- **4 marks** For your level of contribution. We will be using your github repo and your demo to assess this.

5 The Task

You are given a code base that is based on a website to track various uses and their media items.
We have a media server with some dummy data and some functionality removed. You must add the functionality back in and add some new functionality of your own.

5.1 Media Server

The media server is designed to keep track of files and metadata information regarding various audio and video media.
As per the ERD, you have various consumable media:

- Movies
- TV Shows (which have TV Episodes)
- Songs (which are performed by Artists and appear in Albums)
- Podcasts (which have Podcast Episodes)

There is also some User Account information which contains information about the user including:

- Contact Methods
- Username / password
- Subscribed podcasts
- Consumed media

We have abstracted most of the common elements such descriptions, artwork and genres into a new **MetaData** table system.

5.2 SQL Queries x10

Your first task is to complete the 10 missing SQL queries in **’database.py’**
They are as follows:

1. A user logs into the system by providing their username and password. Once logged in, they reach the landing page which displays:
    (a) User subscribed Podcasts
    (b) User Playlists
    (c) User current in-progress items
2. When a user clicks on a song, they should see all the song information, including:
    (a) Song Name
    (b) Song Artists
    (c) Song length
    (d) Song Metadata such as Artwork, description, Genres
3. When a user clicks on the ’tv shows’ nav item, they should see a list of all tv shows information, including:
    (a) TV Show ID
    (b) TV Show Name
    (c) Total number of tv show episodes for this tv show
4. When a user clicks on a single tv Show, they should see a list of relevant information, including:
    (a) TV Show Name
    (b) TV Show Metadata such as Artworks, Descriptions, Genres
    (c) A list of every episode for this tv show ordered by Season and then Episode including:
        i. TV Show Episode ID
        ii. TV Show Episode Title
        iii. Season
        iv. Episode
        v. AirDate
5. When a user clicks on a single Album, they should see a list of relevant information, including:
    (a) Album Name
    (b) Album Metadata such as Artworks, Descriptions
    (c) A list of all songs in an album ordered by track number including:

        i. the song ID
        ii. the Song Name
        iii. The Song Artist(s)

6. Users should also be able to see all the genres for an Album in part (5). The Genres for an album are composed of all the Genres for it's songs.
7. When a user clicks on a single podcast from the 'podcasts' list (or their subscribed podcasts), they should see a list of relevant information including:
    (a) Podcast ID
    (b) Podcast Name
    (c) Podcast URI
    (d) Last Updated
    (e) Podcast Metadata such as Artworks, Descriptions, Genres
    (f) A list of all podcast episodes in this podcast ordered by descending publication date including:
        i. Podcast Episode ID
        ii. Podcast Episode Title
        iii. Podcast Episode URI
        iv. Podcast Episode Date Published
        v. Podcast Episode Length
8. When A user clicks on a single podcast episode from the list in part 7, they should see all relevant podcast episode information including:
    (a) Podcast Episode ID
    (b) Podcast Episode Title
    (c) Podcast Episode URI
    (d) Podcast Episode Date Published
    (e) Podcast Episode Length
    (f) Podcast Episode Metadata such as Artworks, Descriptions, Genres
9. Write the SQL to ensure the proper insert of a new Song, including valid artist checks and appropriate MetaData inserts. You may need to complete the function **addSong()** in the **schema.sql** file and reload the schema.sql file (or just run just that create statement)
10. Write SQL for getting all relevant details for searching through all movies by title

5.3 Data handling tasks x4

Your second task is to complete the data handling for parts (7), (8), (9), (10) from Section 5.2.

You must complete the following functions in **’routes.py’**:

1. **single_podcast(podcast_id)** matching the functionality in 5.2 (7)
2. **single_podcastep(media_id)** matching the functionality in 5.2 (8)
3. **add_song()** matching the functionality in 5.2 (9)
4. **search_movies()** matching the functionality in 5.2 (10)

You can write as many helper functions to assist you as you wish, just remember to document them in your report.

5.4 Data Representation tasks x4

Your third task is to complete the data representation for parts (7), (8), (9), (10) from Section 5.2.

You must complete the following files in the **’templates’** folder:

1. **’templates/singleitems/podcast.html’** matching the functionality in 5.2 (7)
2. **’templates/singleitems/podcastep.html’** matching the functionality in 5.2 (8)
3. **’templates/createitems/createsong.html’** matching the functionality in 5.2 (9)
4. **’templates/searchitems/search_movies.html’** matching the functionality in 5.2 (10) and add a link to **’templates/top.html’** enabling the drop-down option to search by movies

5.5 New Functionality

Your fourth task is to create a new functionality. This will usually involve the following:

1. Create a new function(s) in ’routes.py’ to handle the new route and handle data flow.
2. Create a new function(s) in ’database.py’ or new functions in SQL.
3. Create a new template(s) in ’templates/*.html’.
4. Add new Menu items to ’templates/top.html’ or links in other existing files to access your new routes

We have left some significant gaps between what the schema allows and our current implementation. You should have plenty of space to innovate.
Please try to avoid just copying an existing functionality - e.g. we have given you a search example and an add single item example, do not just iterate out those options to cover all possibilities.
If you find that you can not think of a new feature, then you can implement any number of iterations of existing features instead but we will only give them 0.4 marks each (so you would have to do 5 to get the full 2 marks)

5.5.1 Some ideas to implement

Here are some ideas on what you work on:

- **User Management and security** We do not have any user management implemented. A sample implementation would include **all** of the items below:
  - Secure password storage using at least a hashing and salting technique instead of the raw password (you have access to the pgcrypto extension in the university servers for this functionality)
  - Current User Change Password
  - Current User Display Contact Details on Landing page or in bottom/top bar when logged in
  - Current User Add/Change Contact Details
- **Super User:** Some users are super users. Give them delete access on all records - add a 'delete' button functionality to all single item display, e.g. anywhere a song, movie, podcastep or tvep is listed
- **Multi-term search functionality:** A search that would interpret multiple attributes e.g 'Movie:Once year:>2000'. Can be implemented as it's own page or in the normal search bar.
- **Link Related Content:** Add a link to all instances of Genre so that you would list all MediaItems of the same type that share that Genre. Or using entities with multiple Genre's, Create a display page that would report and add Genre associations.
- **Playlist facility:** Add the ability to create a playlist. A sample implementation would include:
  - Create Playlist page
  - View a Single playlist
  - a button next to any media item in a list or any display page that would allow you to add it to a playlist

Or if you feel like those are too easy, some more interesting but advanced topics that would require some javascript as well:

- **Fuzzy Search:** Upgrade all search/text bars with a fuzzy search / auto-complete search
- **Integrated Webplayer:** While we will be giving you some random user media consumption data, it would be really cool if we could generate some real data. Some of our URI links will lead to valid public domain data like 30s music snippets. Playing said data and monitoring progress would involve integrating a media player into your website and getting feedback from the player to identify the progress a user has made on a particular item. You should also have the option the resume the media automatically from the last point of progress.

5.6 GitHub Issue tracking

As part of your assessment is to determine contributions, we recommend that you use GitHub's issue tracking feature to create and assign issues for each task or subtask. This way it will be easier to track who does what workload.
We will provide further details and links in 5.7

5.7 Scenario Clarification

We will be posting updates to the code base and **sample data** to help you along the way. These notifications will be repeated and collated in this clarification post.
Please refer to updates on the following Edstem post:
https://edstem.org/courses/4680/discussion/326596
with details regarding any clarifications to this specification. This post will be collate every Assignment 03 related clarification every 48 hours. Any clarifications after the 11:59pm Nov 13th 2020 will not be part of the marking specification.

6 Penalty Guidelines

The following contains the penalty guidelines for this assignment. Note that we are extending our penalty guidelines.

6.1 Late Penalty

A lateness penalty of 5% (0.75 marks of your total assessment) per day on your maximum mark will be imposed.

E.g. If your assessment scores 14/15 and you submit 2 days late, your maximum mark will be 15 - 2*0.75 = 13.5/15 so your corrected mark will be 13.5/15.

6.2 Group Participation

Group participation is a requirement for this unit of study and a necessary skill to have.

Students who do not participate in a group and do the assignment themselves will have a **capped mark of 12/15** for this assignment.

We advise students use Issue Tracking on GitHub as outlined in 5.6 as a way to prove their participation and contributions.