CpSc 3720: Team Project Phase 3
Construction

Overview
In this phase of your team project, your team will implement selected functionality and produce related documentation.

Collaboration Policy: You are encouraged to collaborate with others on your team. You must not receive help from anyone not on your team.

Responsibility Assignments
Each team member must take primary responsibility for one coding assignment and one documentation assignment as follows:

Documentation Assignments
1. Screencast and Final Report (1 team member)
2. Software Architecture Document (3 team members)
   
   Note for Teams of 3: This document is to be done by two team members; do a ‘lite’ SAD with no diagrams and only the following sections: Logical, Process, and Implementation views.

Coding Assignments
1. Authenticate User use case
2. Browse Library use case
3. Browse Favorites and Play Track use cases
   
   Note: Play Track will be tested only in conjunction with Browse Favorites.
4. Update Settings use case
   
   Teams of 3 will omit this use case.

Coding Work
Revise your phase 2 work according to grading feedback. Then, enhance it to implement the use cases listed above. Ensure that your design conforms to the architectural standards discussed in class.
Organize code into packages that reflect the logical layers of your design.

Note:
- Functionality that the instructor cannot verify through testing will not receive credit.
- Your applications will be tested on Ubuntu workstations in McAdams 110.
- Every significant method should be commented with a brief header comment.
- No unit tests are needed for this phase.
Software Architecture Document

Create a Software Architecture Document that documents the system from a technical perspective. Follow the format given in class. Include the following:

- **Logical View**
  - Describe the application’s layers
  - Discuss the package structure, highlighting key classes / methods
  - Include a Design Model showing the significant classes and their relationships
  - Include a Sequence Diagram showing how one key method or use case scenario is implemented

- **Process / Deployment View**
  - Briefly describe a typical deployment of your application in an environment with a UMS server. Then, discuss in some detail the communication of your application with the UMS server, including a sample SOAP request and response. Also, discuss the class that handles the SOAP communication for your application, and provide a commented list of the class’s public method signatures.

- **Data View**
  - Include a revised Domain Model diagram.
  - Briefly describe each class on the model.

- **Security (Omit)**

- **Implementation**
  - Discuss project directory structure and key source code files

- **Development**
  - Provide detailed step-by-step instructions for a new developer to check out the code, build and test from command line, and set up and run the project in an IDE. Include instructions on running the UMS server.

- **Use Case**
  - Provide a Use Case diagram showing the implemented use cases.
  - Copy and paste the use case descriptions from the Development Plan into this section. Modify the descriptions as needed to indicate the functionality actually implemented.

Note 1: Due to the number of diagrams and potentially complex formatting in this document, you may use a word processor to create this and publish it as a PDF, instead of a regular wiki page.

Note 2: If you wish your work to be graded individually, put your name on each portion of the document that you did. Otherwise, all team members assigned to this responsibility will receive the same grade.

Screencast and Issues Report

Create a short (4 minutes or less) screencast illustrating the functionality.

Also, create a wiki page named Final Report listing known issues and two links: 1) link to the screencast, 2) link to the root folder of the final project in the GitHub repo. Carefully review the requirements in the Development Plan and test the application when preparing this report. Significant issues that are undocumented will result in point deductions off this portion of the deliverable.
Code Deployment

- Your project must contain bash scripts in the root of the project folder, named gobuild and goapp, as in previous deliverables.
- Include in the repository a settings file and locally downloaded media that reflects the following scenario, so that when your app is run these settings are in effect, and media can be played from downloaded favorites without having to browse/download from the media server:
  - Server URL: http://127.0.0.1:5001/upnp/control/content_directory
  - User Profiles:
    - Admin (PIN 9999), No favorites
    - Child 1 (PIN 1111), Restriction Level 2, Favorites: Album A
    - Child 2 (PIN 2222), Restriction Level 3, Favorites: Album A, Album B
  - 3 Restriction Levels: Level 1 = Album A; Level 3 = Album B, Album C

Submission

- Deadlines:
  - Coding Deadline: All coding work must be completed by 11:59pm, Thursday, March 3. Code changes are not allowed after this deadline.
  - Documentation Deadline: Documentation is due by 2:00pm, Tuesday, March 8.
- The Home page of the wiki must link to the Final Report and Architecture Document, list each team member’s name, and clearly indicate which work items each team member is responsible for, per the Responsibility Assignments above.

Grading

Team members will be graded using the following rubric.

<table>
<thead>
<tr>
<th></th>
<th>Excellent</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness / Visual Design</td>
<td>20 points</td>
<td>17 points</td>
<td>14 points</td>
<td>5-10 points</td>
</tr>
<tr>
<td></td>
<td>All functionality correctly implemented; UI incorporates Phase 2 feedback</td>
<td>minor problems</td>
<td>Effort shown, but significant shortcomings</td>
<td>Minimal effort shown</td>
</tr>
<tr>
<td>Code Style / Design</td>
<td>15 points</td>
<td>13 points</td>
<td>11 points</td>
<td>5-9 points</td>
</tr>
<tr>
<td></td>
<td>Good code architecture; code well commented</td>
<td>minor problems</td>
<td>Effort shown, but significant shortcomings</td>
<td>Minimal effort shown</td>
</tr>
<tr>
<td>Documentation</td>
<td>10 points</td>
<td>8 points</td>
<td>6 points</td>
<td>4 points</td>
</tr>
<tr>
<td></td>
<td>Comprehensive content, professionally formatted</td>
<td>minor problems</td>
<td>Significant shortcomings</td>
<td>Minimal effort shown</td>
</tr>
<tr>
<td>Team Submission</td>
<td>5 points</td>
<td>4 points</td>
<td>3 points</td>
<td>1 point</td>
</tr>
<tr>
<td></td>
<td>Build / launch scripts work; Home page contains required info</td>
<td>Minor submission issue</td>
<td>Significant submission issue requiring instructor email</td>
<td>Serious submission issues</td>
</tr>
</tbody>
</table>

Note: Code Style will be graded only if Correctness receives a rating of at least “Fair.”