JEE

Mini Project

Media Composer and Artist

Management System

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| ***Reference No:*** |  | | |
| ***Security Classification:*** | **Internal** | | |
| ***Issue Date:*** |  | | |
| Author(s): | **Name** | | **Title** |
| Rathnajothi | | Senior Executive – T & D |
|  | |  |
| Reviewer(s): | **Name** | | **Title** |
| Training Team | |  |
| Issuer(s): | **Name** | | **Title** |
|  | |  |
| Issuer Signature(s): |  | | |
| Distribution: | **Name** | | **Title** |
| Team | | All Designations in Training |
|  | |  |
|  | |  |
| Document History: | **Date** | **Revision** | **Change** |
| 01-08-2014 | 0.01D | Initial draft |
| 29-06-15 | 0.02D | Changes in Miniproject document content as per the upgraded courses |
|  |  |  |

Table of Contents

[1 Introduction 4](#_Toc394783426)

[1.1 Setup Checklist for Mini Project 4](#_Toc394783427)

[1.2 Instructions 4](#_Toc394783428)

[2 Problem Statement 5](#_Toc394783429)

[2.1 Objective 5](#_Toc394783430)

[2.2 Abstract of the project 5](#_Toc394783431)

[2.3 Functional components of the project 5](#_Toc394783432)

[2.4 Technology used: 6](#_Toc394783433)

[3 Implementation in J2EE LOT 7](#_Toc394783434)

[3.1 Summary of the functionality to be built: 7](#_Toc394783435)

[3.2 Guidelines on the functionality to be built: 8](#_Toc394783436)

[3.3 Evaluation and assessment parameters: 11](#_Toc394783437)

# Introduction

This document outlines a mini project for the J2EE LOT. The project is to develop an Media Composer and Artist Management System for an organization. This document contains the work flow of the system and gives guidelines on how to build the functionality gradually in each of the course modules of the J2EE LOT.

## Setup Checklist for Mini Project

Minimum System Requirements

* Intel Pentium 90 or higher (P166 recommended)
* Microsoft Windows 95, 98, or NT 4.0, 2k, XP, Windows 7
* Memory: 32MB of RAM (64MB or more recommended)
* Oracle 9i client and access to oracle 9i server
* JDK 8
* Eclipse Luna
* JUnit 4.0

## Instructions

* The code modules in the mini project should follow all the coding standards.
* Create a directory by your name in drive **<drive>**. In this directory, create a subdirectory **MiniProject**. Store your Project here.
* You can refer to your course material.
* You may also look up the help provided in the java docs and documentation provided with WildFly.
* The total time required to complete this mini project is 50 hrs.
* Since this project work will span over couple of months, you will need to take care of maintaining the code

# Problem Statement

## Objective

Development of an Media Composer and Artist Management System.

## Abstract of the project

This project is aimed at developing an Media Composer and Artist Management System. application that facilitates searching and adding of new composers and artists related to songs. Artists and composers are performers for a specific master of a song. The system must enable the business to enter information about artists or composers to associate with masters of songs in order to later perform search functions for specific recordings. Artists may be an individual person/ group of persons.

## Functional components of the project

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate, you can make appropriate assumptions and proceed.

There are two categories of people who would access the system viz. Admin & User. Each one of them would have some exclusive privileges (for e.g. Admin can create Artist/Composer, edit Artist/Composer details, Associate song(s) to composer/artists, search for composer/artist).

1. Admin should be able to
   * Login to the system using his/her credentials
   * Search for composer
   * Add new composer
   * Edit an existing composer details.
   * Search for artist.
   * Add new artist.
   * Edit an existing artist details.
   * Associate song(s) to an artist/composer.
2. The User should be able to
   * Login into the system using his/her credentials.
   * Search for a song based on composer/artist details.

## Technology used:

* + - *Front End & Web Components:–* 
      1. Java Classes
    - *Business Logic Components and Services :-* 
      1. Java Beans
    - *Databases:-*
      1. Oracle 9i

# Implementation in J2EE LOT

## Summary of the functionality to be built:

The participants need to develop the Media Composer and Artist Management System by building the functionality incrementally in each of the course modules of J2EE LOT.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Course** | **Duration**  **(in PDs)** | **No. of Saturdays** | **Functionality to be built** |
| 1 | Programming Foundation with Pseudo code | 3 | 1 | Analyze the given case study |
| 2 | Introduction to Software Engineering | 0.5 |  | Analyze the Case study using SDLC phases. |
| 3 | Web Basics (HTML 5,CSS 3, JavaScript, XML) | 4.5 | 1 | Developing prototype i.e. developing screens/web pages in HTML and client side validation in JavaScript. |
| 4 | Oracle Basics | 4 | 1 | Creating relevant database tables |
| 5 | OOP & UML | 1.5 | 1 | Creating relevant Use case and class diagrams |
|  | Programming Foundation with Pseudo code + Web Basics +Oracle Basics +OOP & UML Test | 1 |
|  | Core Java 8 & Development Tools (Junit, Log4j) | 10 | 2 | Developing Business components (java classes). Coding for test classes & testing the functionality using JUnit |
|  |  |  |  |
|  | Core Java 8 + Dev Tools + OOP/UML Test | 1 |  |
| 8 | Servlets | 3.5 | 2 | Project specific implementation not needed as mini project is in Core Java |
| 9 | JSP | 2 |
| 10 | Developer Workbench (PMD, MAVEN) | 1 |
| 11 | Servlets + JSP + Dev Workbench Test | 1 |
| 12 | Basic Spring 4.0 | 5 | 1 | Prepare document for presentation. |
| 13 | Basic Spring Test | 1 |
| 14 | Mini Project presentation | 1 |  |  |

## Guidelines on the functionality to be built:

The functionality and components to be built in each of the course modules of J2EE LOT is as follows: Convert your HTML Pages into Console Screen using Java class with below options.

1. Course: HTML, JavaScript **(Duration: 10 hours)**
   1. Develop the following screens:
      1. Home : Home page for Media Composer and Artists System which provides option for the login page.
      2. Login : Allows the valid user/surveyor/respondent to logon to the system and display the Main option screen.
      3. Main option screen: For Admin, this screen will display options for to create Artist/Composer, edit Artist/Composer details, Associate song(s) to composer/artists, and search for composer/artist. For User, this screen will allow the user to search a song based on composer/artist details.
      4. Create Composer screen: This screen allows the admin to create Composer for a song.
      5. Create Artist screen: This screen allows the admin to create Artist for a song.
      6. Edit Composer screen: This screen allows the admin to edit Composer details.
      7. Edit Artist screen: This screen allows the admin to edit artist details.
      8. Associate song(s) to composer/artists screen: This screen allows the admin to associate a song/list of songs to a composer/ artist.
      9. Search for composer/artist: This screen allows the admin to search for a composer/artist.
      10. Navigate to home page at any point of time
      11. Logoff from the application at any point of time
   2. In this course you need to develop the user interface using Java classes for console screen and document the flow of your application including the images of console screen in a word document. The screens should include the fields as per the functionality mentioned above. Also, include validations using Core java regular expression in each of these screens
2. Course: Oracle **(Duration: 5 hours)**
3. Create the following database tables:
   * 1. User\_Master - To store USER login details
     2. Composer\_Master - To store the Composer details
     3. Song\_Master - To store the song details
     4. Artist\_Master - To store the artist details
     5. Artist\_Song\_Assoc - To store the artist and the related songs
     6. Composer\_Song\_Assoc - To store the composer and the related songs
     7. MusicSociety\_Master - To store the society details
4. The structure of the above listed tables is as follows:
   * 1. **User\_Master:** User\_ID NUMBER(6),User\_Password VARCHAR2(20),Created\_By NUMBER(6),Created\_On Date,Updated\_By NUMBER(6),Updated\_On Date.

For Admin and User assume that the users are already added to the system.

* + 1. **Composer\_Master:** Composer\_id NUMBER(6), Composer\_Name VARCHAR2(50), Composer\_BornDate Date, Composer\_DiedDate Date, Composer\_CaeipiNumber VARCHAR2(10), Composer\_MusicSocietyID CHAR(3), Created\_By NUMBER(6), Created\_On Date,Updated\_By NUMBER(6), Updated\_On Date, Composer\_DeletedFlag Boolean
    2. **Song\_Master:** Song\_Id NUMBER(6), Song\_Name VARCHAR2(50), Song\_Duration TIMESTAMP, Created\_By NUMBER(6), Created\_On Date, updated\_By NUMBER(6), Updated\_On Date, Song\_DeletedFlag Boolean

* + 1. **Artist\_Master:** Artist\_Id NUMBER(6), Artist\_Name VARCHAR2(50), Artist\_Type CHAR(1), Artist\_BornDate Date,Artist\_DiedDate Date, Created\_By NUMBER(6),Created\_On Date, Updated\_By NUMBER(6), Updated\_On Date, Artist\_DeletedFlag Boolean
    2. **Artist\_Song\_Assoc:** Artist\_Id NUMBER(6), Song\_Id NUMBER(6), Created\_By NUMBER(6), Created\_On Date, Updated\_By NUMBER(6), Updated\_On Date
    3. **Composer\_Song\_Assoc:** Composer\_Id NUMBER(6), Song\_Id NUMBER(6), Created\_By NUMBER(6), Created\_On Date, Updated\_By NUMBER(6), Updated\_On Date
    4. **MusicSociety\_Master:** Composer\_MusicSocietyID CHAR(3), Composer\_MusicSocietyName VARCHAR2(50)

Note: You may add/normalize/denormalize the tables if your application demands it.

1. Course: OOP & UML **(Duration: 5 hours)**
2. Develop relevant Use case and Class diagrams for the application.
3. Course: Core Java 8 + Developer Tools **(Duration: 14 hours)**
4. Develop business components (java classes) for the following functionality:
   * 1. Authentication Service (on Login): This component will verify if the user who is trying to access the system is a valid user. This verification is as against the valid users listed in the User\_Master table.
     2. Admin Service: This component will allow the admin to create Artist/Composer, edit Artist/Composer details, Associate song(s) to composer/artists, and search for composer/artist.
     3. User Service: This component will allow the user to search for a song based on composer/artist details

ss

* 1. Develop test classes for testing the following functionality and Test the application using JUnit
     1. Login
     2. Create composer
     3. Edit artist details
  2. Configure Logger to log the status of an application

1. Course: Core Java Clases + Developer Workbench **(Duration: 14 hours)**
2. Convert all the java classes (business components) created in Java module to Java beans
3. Integrate all screens with business components (java beans) to complete the entire functionality
4. Documentation **(Duration: 2 hours)**
   1. Project Documentation: Document your project details (Duration: 1 hour 30 min’s).
   2. Project submission: Submit your project with all the artifacts including the test cases & documentation (Duration: 30 min’s).

Application Architecture: Discuss this with your mentor on regular basis.

## Evaluation and assessment parameters:

This mini project will be done in groups of five. Each group will identify a Team Lead who will decide which team member will code for which functionality. This project shall be evaluated at the end of spring module.

**Evaluation Criteria (out of 100):**

|  |  |
| --- | --- |
| Look of console for all the screens | **05** |
| Client-side validation of inputs | **10** |
| Code Documentation and using coding standards | **10** |
| Overall Business logic. This includes:   * Usage of Logging API (log4j) | **30** |
| Good amount of appropriate dataset to showcase project completely | **5** |
| Appropriate test cases using JUnit 4.0 | **5** |
| Using MVC architecture and clean encapsulation of business logic in appropriate components. Judicious use of java beans. | **35** |