**A Project Report**

###### ***Submitted by***

##### Tirtha Chanda

Shreya Das

Subham Nandy

**Online Music Library**

***In the fulfillment for the award of the degree of***

##### **BCA**

**at**

**Techno India**

**Feb-May 2016**

**Certificate From Supervisor**

This is to certify that Mr. Tirtha Chanda, Miss Shreya Das, and Mr. Subham Nandy successfully completed the project titled **"Online Music Library**" under my supervision during the period from February to May which is in partial fulfillment of requirements for the award of the Bachelor in Computer Applications.

*Signature and Seal of the Supervisor*

**Date:**  **Name & Designation**

**Project Supervisor**

**Acknowledgement**

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, Munmuni Mukherjee for giving such valuable suggestions, guidance and encouragement during the development of this project work.

|  |  |
| --- | --- |
| **Contents** | **Page No** |
| 1. Abstract | 6 |
| 1. Introduction | 6 |
| 1. Project Category | 6 |
| 1. Tools/Platform | 6 |
| 1. Methodology (SDLC process applied) | 7 |
| 1. Functional Requirements | 8 |
| 1. Non-Functional Requirements | 8 |
| 1. System Analysis | 9 |
| 1. Project Planning | 19 |
| 1. Project Scheduling | 19 |
| 1. System Design | 20 |
| 1. Coding | 22 |
| 1. Testing | 25 |
| 1. Contribution of the work | 31 |
| 1. Future scope | 31 |
| 1. Conclusion | 31 |
| 1. Bibliography | 32 |

1. **Abstract**

A huge collection of songs, the ability to create playlists based on a persons' interest, and the option to download any song as per users' choice are some of the features provided by the system. An admin may update the system and upload new songs and remove already uploaded songs. The online music library will provide the user full access to a limitless collection of songs in English, Hindi, and Bengali languages.

**2. Introduction**

Online Music Library System is a specific requirement of the client that integrates playing and downloading music online

* Need for the new system is due to major drawbacks of the existing system
* Reports can be generated at any time within a few seconds, so that manual labour is not required. Also, analysis can be performed much more frequently.
* The details regarding all users, songs can be maintained.
* Allows users to register from their homes via a web browser.

**3. Project Category**

Web Application

**4. Tools/Platform**

1. Eclipse MARS2
2. Microsoft Windows 7/8/10
3. J2EE
4. Oracle Database
5. Apache Tomcat 8

**5. Methodology (SDLC Process Applied)**



Often, a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.

The prototyping paradigm begins with **requirements gathering**. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A **"quick design"** then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

Ideally, the prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to use existing program fragments or applies tools (e.g., report generators, window managers) that enable working programs to be generated quickly.

**6. Functional Requirements**

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

**7. Non Functional Requirements**

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually DO anything, but are important characteristics nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility. aren't a "feature" of the system, but are a required characteristic. You can't write a specific line of code to implement them; rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in English. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making tradeoffs, e.g., between performance and maintainability. In the APPENDIX you must include an engineering analysis of any significant decisions regarding tradeoffs between competing attributes.

Here are some examples of non-functional requirements:

**Performance requirements**

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

For better performance the application will restrict the document size to 5 MB.

**Operating constraints**

List any run-time constraints. This could include system resources, people, needed software,

The application must run without any manual intervention.

**Platform constraints**

Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

Since the application will be developed in JEE it is platform independent.

**Accuracy and Precision**

Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

**Modifiability**

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Minimal

**Portability**

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

**Reliability**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

**Security**

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time) to break into the system.

Only secured users can access the application.

No one can go to any independent page without logging in.

**Usability**

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

**Legal**

There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies

**8. System Analysis**

**8.1 Feasability**

**Feasibility test**

One should provide a feasibility report in the following format:

* **Product:**A general statement of the product; give a brief description of what the proposed system will do, highlighting where the proposed system meets the specified business requirements of the organization.
* **Technical Feasibility:**Will the proposed system perform to the required specification? Outline technical systems options you propose to use, which will give a technical solution satisfying the requirements and constraints of the system, as outlined in the terms of reference.
* **Social Feasibility:**Consideration of whether the proposed system would prove acceptable to the people who would be affected by its introduction. Describe the effect on users from the introduction of the new system; consider whether there will be a need for retraining the workforce. Will there be a need for relocation of some of the workforce? Will some jobs become deskilled? Will the current workforce be able to perform effectively any new tasks introduced by the proposed system? Describe how you propose to ensure user co-operation before changes are introduced.
* **Economic Feasibility:**Consider the cost/benefits of the proposed system. Detail the costs that will be incurred by the organization adopting the new system; consider development costs and running costs. Detail benefits that the new system will bring, direct economic benefits such as reduced costs, and indirect benefits, such as improved management information and better customer service. Illustrate the cost/benefit of the new system by applying a suitable cost/benefit analysis method such as the payback method.
* **Market Research:**A comprehensive market research identifying a need for the product. Detail all market research you carried out, listing sources of information. Justify any conclusions you have drawn from your research. Identify the potential customer base for your product, together with evidence of customer need for the product. Describe how you propose to compete with similar products on the market.
* **Alternative Solution:**Consideration of alternative solutions should be documented. At least two alternative business or technical systems options should be considered. Detail the differences between these options and the proposed system. Justify your choice of the proposed system and the reasons for rejecting the alternative options.

At this point, all of the planning for the project has been done and if the feasibility study has shown that the project is likely to succeed within its constraints, then it only remains for us to start the requirements analysis and thus proceed with the project.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Feasibility Study** | | | System: Online Music Library System | Date: 20/2/2016 | | Author: SPMS | Page: 1 | | **Product** | | | The project requires a web application to be developed that will allow online knowledge/document/paper sharing. | | | **Technical Feasibility** | | | The web application will be developed using J2EE, HTML, CSS and Oracle. | | | **Social Feasibility** | | | Some training for the Administrator and EO are required. | | | **Market Research** | | | Market research says that this application would be useful for the users as it could seamlessly help them to share documents. | | | **Economic Feasibility** | | | The application can be developed within budget. | | | **Alternate Solution** | | | Could be a desktop system but that would not allow documents to be shared online. | | | |
|  |  |
|  |  |

**8.2. Existing System**

At present there is no online system for music in our country. People have to go buy music CDs.

**8.3. Proposed System**

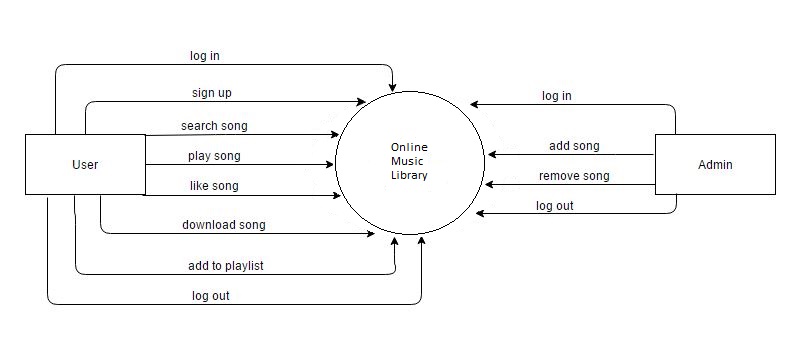
The aim of the project is to make an online application that will help users play and download music online, and also create custom playlists.

**8.4. Data Flow Diagram**

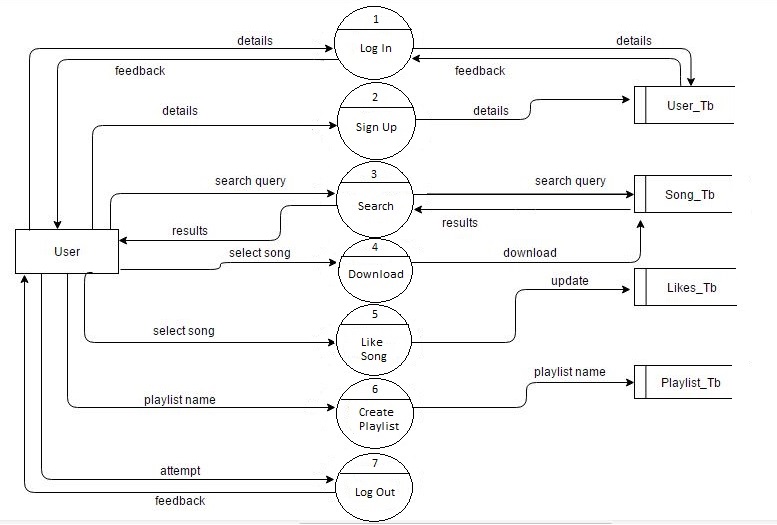
A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFD can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

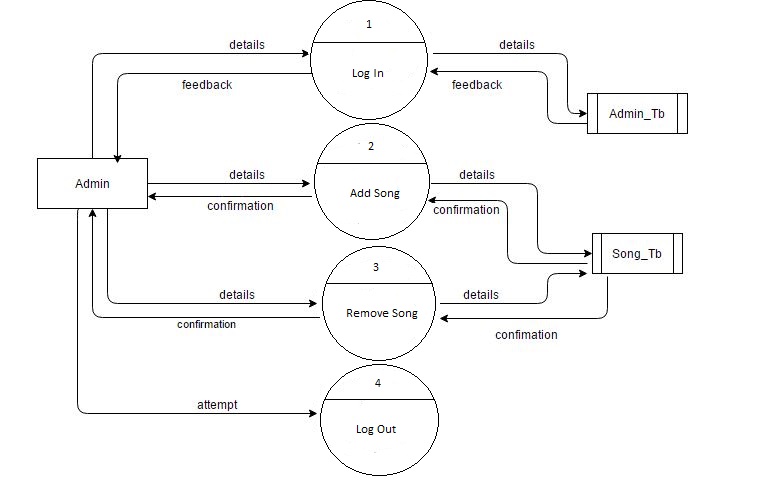
**Level 0 DFD**

****

**Level 1 DFD (User)**

****

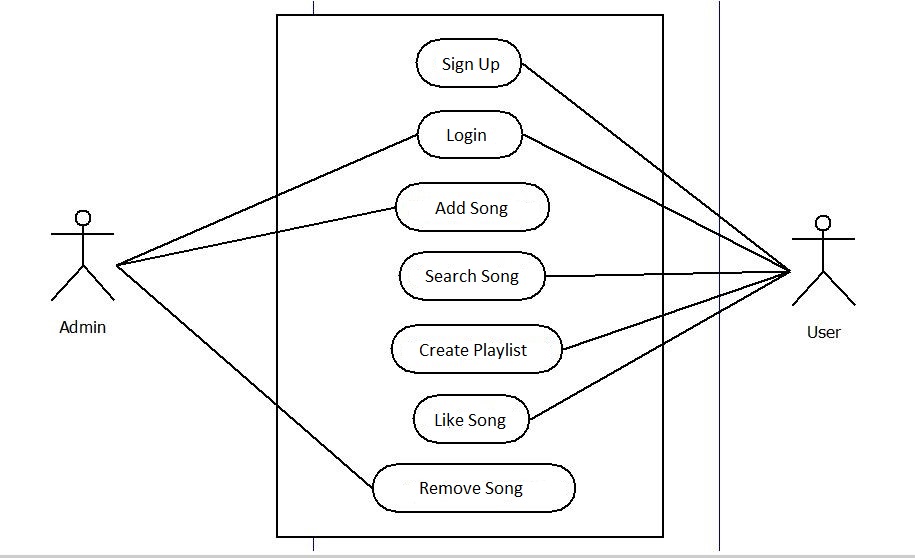
**Level 1 DFD (Admin)**

****

**8.5 Use Case Diagram**

A use case in software engineering and systems engineering is a description of steps or actions between a user ( “actor") and a software system which lead the user towards something useful. The user or actor might be a person or something more abstract, such as an external software system or manual process. Use cases are a software modeling technique that helps developers determine which features to implement and how to gracefully resolve errors.

Each use case focuses on describing how to achieve a goal or a task. For most software projects, this means that multiple, perhaps dozens of use cases are needed to define the scope of the new system. The degree of formality of a particular software project and the stage of the project will influence the level of detail required in each use case.

****

The requirements of a system can be represented using a use case model in the Use Case Diagram. The use case diagram for the actors of this case study is given as below.

**Log in**

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | *Login* | |
| **Goal** | *All users logging into the system should be authenticated using a unique login-id and password (operations to be supported based on type of user)* | |
| **Preconditions** | If the user type is ‘Admin,’ credential details should exist.  If the user is ‘Customer,’ they can log in or sign up | |
| **Success End Condition** | If the user type is ‘Admin’, then redirect to the Admin page.  If the user type is ‘Customer, then redirect to the Customer home page. | |
| **Failed End Condition** | *The end user is redirected to an Error Page, and/or is asked to reenter login credentials.* | |
| **Primary, Secondary Actors** | Admin, Customer | |
| **Trigger** | *Login button* | |
| **DESCRIPTION** | **Step** | **Action** |
| **1** | *Enter Login credentials (id & password)* |
| **2** | *Click on Login button* |
| **3** | *If id & password is Success, then identify user type Display appropriate(Admin/EO/voter) home page* |
| **Step** | **Branching Action** |
| **1** | *If 'id' is not existing then return with requesting for registration* |
| **2** | *If password is not matching return with suitable error message say 'reenter id & password'* |
| **Related**  **Information/Use cases** | *Not Applicable* | |
| **Priority** | *P1* | |
| **Performance** | *Approx. in seconds* | |
| **Frequency** | *Approx. few users per minute* | |
| **Assumptions** | Admin/Customer login credentials are available in the database and others are already registered with their credentials | |

Below table explains ‘Use Case’ definition for requirement “Create Playlist”

**Create Playlist**

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | *Create Playlist* | |
| **Goal** | *To enable User to create a new playlist* | |
| **Preconditions** | User must be logged in to be able to create a new *playlist*. | |
| **Success End Condition** | *“Redirect to Playlist page”* | |
| **Failed End Condition** | *"Redirect to Error Page"* | |
| **Primary, Secondary Actors** | *User* | |
| **Trigger** | *Create Playlist button* | |
| **DESCRIPTION** | **Step** | **Action** |
| **1** | *Provide playlist name* |
| **2** | *Click on Create button* |
| **Step** | **Branching Action** |
| **1** | *If failed to create* |
| **2** | *Display appropriate message to the User* |
| **Related**  **Information/Use cases** | *Not applicable* | |
| **Priority** | *P1* | |
| **Performance** | *Approx. 0.5 sec* | |
| **Frequency** | *400 / month* | |
| **Assumptions** | User login credentials are available in the database | |

**8.6. Software Requirements**

**Software Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **Browser** | Any standard browser with Javascript interpreter | **Software** | Eclipse MARS2 |
| **Client side mark up / scripting languages** | HTML,CSS, Javascript | **Database Management System Software** | Oracle 11g |
|  |  | **System Software** |  |
|  |  | **Specification** | J2EE |

**9.6.1 Front End:** We have designed the front end using html and css.

**9.6.2 Middle End:** We have used Eclipse MARS2 as a middle end.

**9.6.3 Application Server:** We have used Apache Tomcat 8 as our application server.

**9.6.4 Database Server:** We have used Oracle 11g as our database server.

**8.7. Hardware Requirements**

**Hardware Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **HDD** | 200 MB | **HDD** | Minimum 30 GB. |
| **Processor** | Pentium 4 or newer processor that supports SSE2 | **Processor** | Dual Core or newer processor |
| **Memory** | 512 MB | **Memory** | GB |

**9. Project Planning**

Project planning is concerned with identifying the following for every project:

* Activities
* Milestones
* Deliverables.

A plan must be drawn up to guide the development towards the project goal. A plan is drawn up at the start of a project. This plan should be used as the driver for the project. The initial plan is not static, and must be modified as the project progresses.

Planning is required for development activities from specification through to delivery of the system.

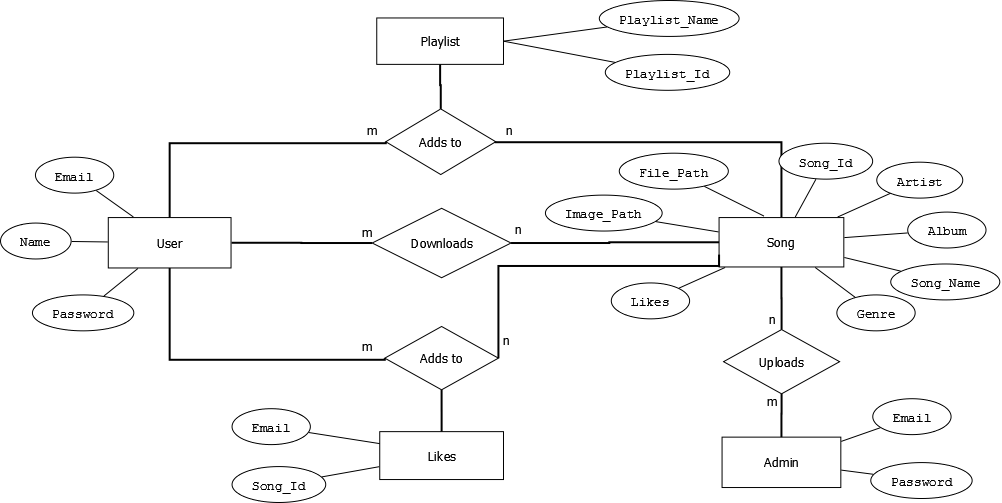
**10. Project Scheduling**

**GANTT chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **Person(s) Responsible** | **Week 1** | **Week 2** | **Week 3** | **Week 4** |
| **Communication** |  |  |  |  |  |
| **Quick Plan** |  |  |  |  |  |
| **Modeling Quick Design** |  |  |  |  |  |
| **Construction of Prototype** |  |  |  |  |  |
| **Deployment, Delivery and Feedback** |  |  |  |  |  |

**11. System Design**

**ER DIAGRAM**



**TABLE DESIGN**

## 1. Table: Admin\_Tb

This table contains Authentication Information for Admins

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Email | VARCHAR2(255) |
| Name | VARCHAR2(255) |
| Password | VARCHAR2(255) |

## 2. Table: User\_Tb

This table contains User specific details entered during User Registration.

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Email | VARCHAR2(255) |
| Password | VARCHAR2(255) |
| Name | VARCHAR2(255) |

## 3. Table: Song\_Tb

This table contains song information.

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Song\_Id | VARCHAR2 (50) |
| Artist | VARCHAR2(255) |
| Album | VARCHAR2(255) |
| Song\_Name | VARCHAR2(255) |
| File\_Path | VARCHAR2(255) |
| Image\_Path | VARCHAR2(255) |
| Likes | VARCHAR2(10) |
| Genre | VARCHAR2(255) |

## 4. Table: Playlist\_Tb

This table contains playlist information.

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Playlist\_Id | VARCHAR2(50) |
| Email | VARCHAR2(255) |

## 5. Table: Playlist\_Songs

This table contains songs currently in a user’s playlist.

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Playlist\_Id | VARCHAR2 (50) |
| Song\_Id | VARCHAR2(50) |

## 6. Table: Likes\_Tb

This table contains a all songs liked by users

|  |  |
| --- | --- |
| **Field Name** | **Data Type** |
| Email | Integer(255) |
| Song\_Id | VARCHAR2(50) |

**12. Coding**

Some code snippets are given below:

**Log In Code:**

/\* request.getParameter is used to get the value from the previous page's form \*/

String email = request.getParameter("email");

String password = request.getParameter("password");

PreparedStatement pst = **null**;

ResultSet rs = **null**;

**try** {

/\* checks admin\_tb for a match \*/

pst = conn.prepareStatement("Select Email, Password From Admin\_Tb Where Email=? and Password=?");

pst.setString(1, email);

pst.setString(2, password);

/\* stores result in rs \*/

rs = pst.executeQuery();

/\*

\*if a row matched (rs.next() returns non zero),

\*creates session id and type variable, and redirects to admin home page

\*/

**if** (rs.next()){

session.setAttribute("id", email);

session.setAttribute("type", "admin");

response.sendRedirect("AddSong.jsp");

}

/\*

\*this is reached if admin table didn't produce a match

\*checks user\_tb for a match

\*/

pst = conn.prepareStatement("Select Email, Name, Password From User\_Tb Where Email=? and Password=?");

pst.setString(1, email);

pst.setString(2, password);

/\* stores result in rs \*/

rs = pst.executeQuery();

/\*

\*if a row matched (rs.next() returns non zero),

\*creates session id and type variables, and redirects to user home page

\*/

**if** (rs.next()){

String name = rs.getString(2);

//splits name by whitespace and keeps only the first name

name = name.split("\\s+")[0];

session.setAttribute("id", email);

session.setAttribute("name", name);

session.setAttribute("type", "user");

response.sendRedirect("UserHome.jsp");

}

**else**{

%><script>

window.alert('Incorrect Email/Password');

window.location.href = "LogIn.jsp";

</script><%

}

}

**catch**(Exception e) {

out.println(e);

}

**Add To Playlist Code:**

String songId = request.getParameter("songId");

String playlistId = request.getParameter("buttonId");

Connection conn = Connect.*getConnection*();

response.setContentType("text/xml");

response.setHeader("Cache-Control", "no-cache");

**try**{

PreparedStatement pst = conn.prepareStatement("Insert Into Playlist\_Songs(Playlist\_Id, Song\_Id) Values(?, ?)");

pst.setString(1, playlistId);

pst.setString(2, songId);

pst.executeUpdate();

response.getWriter().write("<valid>available</valid>");

}

**catch**(Exception e){

System.***out***.println("Add to Playlist Exception: " + e);

response.getWriter().write("<valid>failed</valid>");

}

**Download Song Code:**

String filename = request.getParameter("songName").concat(".mp3");

String filePath = "C:\\My Files\\Code\\J2EE\\Projects\\AudioCloud\\WebContent\\Music\\" + filename;

File downloadFile = **new** File(filePath);

FileInputStream inStream = **new** FileInputStream(downloadFile);

// if you want to use a relative path to context root:

String relativePath = getServletContext().getRealPath("");

System.***out***.println("relativePath = " + relativePath);

// obtains ServletContext

ServletContext context = getServletContext();

// gets MIME type of the file

String mimeType = context.getMimeType(filePath);

**if** (mimeType == **null**) {

// set to binary type if MIME mapping not found

mimeType = "application/octet-stream";

}

// modifies response

response.setContentType(mimeType);

response.setContentLength((**int**) downloadFile.length());

// forces download

String headerKey = "Content-Disposition";

String headerValue = String.*format*("attachment; filename=\"%s\"", downloadFile.getName());

response.setHeader(headerKey, headerValue);

// obtains response's output stream

OutputStream outStream = response.getOutputStream();

**byte**[] buffer = **new** **byte**[4096];

**int** bytesRead = -1;

**while** ((bytesRead = inStream.read(buffer)) != -1) {

outStream.write(buffer, 0, bytesRead);

}

inStream.close();

outStream.close();

**13. Testing**

**Team Interaction**

The following describes the level of team interaction necessary to have a successful product.

* The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.
* The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.
* Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

# 

# Test Objective

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

# The Process Overview

The following represents overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

# Testing Process

**b.** Design System Test

a. Organize Project

e. Design/Build Test Proc.

f. Log Out

d. Organize Project

c. Design/ Build

Fig 18.1

The diagram above outlines the Test Process approach that will be followed.

1. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

**b.** **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

**c.** **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

**d.** **Build Test Environment** includes requesting/building hardware, software and data set-ups.

**e. Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

**f.** **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

# Testing Strategy

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document.

**Test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Tirtha Chanda | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 1 | |
| **Test Case Name** | | Log In | |
| **Test Case Description** | | The user should enter his/ her accurate userid and password so that he/she can opt for the further options. The test case will check the application for the same since a user can only login with the correct userid , password. | |
| **Item(s) to be tested** | | | |
| 1 | Verification of the userid and password with the record in the database. | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| 1. Correct User id and password 2. Incorrect Id or Password | | | 1. Successful login 2. Failure Message |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Shreya Das | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 2 | |
| **Test Case Name** | | Sign Up | |
| **Test Case Description** | | The user fills up the registration form. | |
| **Item(s) to be tested** | | | |
| 1 | Check whether the user has filled up the form properly. | | |
| 2 | Wait for acknowledgement from the server. | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| 1. Trying to submit without filling the form properly. 2. Check for acknowledgement from the server. | | | 1. The user is shown an error message. 2. Redirect accordingly. |

**Unit Testing**

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the program’s correctness.

### Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

## System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focus only on function validation and performance. And in both cases we will use the black-box method of testing.

**14. Contribution of the work**

We have developed this project by distributing the project work among all the team members. Tirtha Chanda has designed the GUI and animations of the website and has also handled the implementation of the user registration and log in, and also the search functionalities. Shreya Das has handled all of the database design and implementation, and also the file upload/download implementation, and assisted with the search implementation. Subham Nandy has coded the JavaScript, database connectivity, and also assisted with the Playlist implementation.

**15. Future scope**

* In future documents may be classified category-wise. It may try to analyze the user behavior and preferences and accordingly suggest products.
* In this project, we will add more modules that support extra information.
* Users will be able to track their packages directly on the website.
* In future we can add the option of sending mail after the successful registration.

**16. Conclusion**

The application has been designed, implemented and tested successfully. The project helped us in understanding the challenges involved in developing an online application, the ways to overcome them. It has also helped in understanding the value of designing the components of overall application before implementing them. In addition to that, it has also taught us programming skills and refining the design and implementation logic of the software at every phase of the development life cycle in order to improve the overall performance of the application. This has also taught us to work in a team. Lastly it has boosted our confidence that if we get a similar type of project in future, we will be able to contribute our knowledge and experience.

**27. Bibliography**

1. Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.
2. Ian Sommerville. Software Engineering (Seventh Edition). Addison-Wesley, 2004.
3. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley Pub Co; 1st edition (August 2, 1995).
4. 1st Edition: 1975. Tells the story of the IBM 360 Operating System, and what failed why.
5. UML = Unified Modeling Language
6. Martin Fowler. UML Distilled. Addison-Wesley Pub Co; 3rd edition (September 19, 2003).
7. Edward Yourdon and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and System Design. Prentice-Hall, 1979.
8. Ken Arnold and James Gosling, The Java Programming Language, second ed., Addison-Wesley, 1998.
9. Gary Cornell and Cay S. Horstmann, Core Java, second ed., SunSoft Press, 2013

[www.oracle.com](http://www.oracle.com)/oracle-tutorial

[www.w3schools.com](http://www.w3schools.com)/css

[www.tutorialspoint.com/c#](http://www.tutorialspoint.com/c)