|  |  |
| --- | --- |
|  | **Cognizant Academy**  **movieCruiser**  **ADO.NET Specification Document**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Ramamoorthy Selvam | Vimalathithan Krishnan | Ramadevanahalli Lingachar, Shashidhara Murthy | | **Role** | Learning Solution Designer | Learning Solution Architect | Learning Solution Lead | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

Table of Contents

[1.0 Introduction 3](#_Toc11949068)

[1.1 Purpose of this document 3](#_Toc11949069)

[1.2 Definitions & Acronyms 3](#_Toc11949070)

[1.3 Project Overview 3](#_Toc11949071)

[1.4 Scope 3](#_Toc11949072)

[1.5 Intended Audience 3](#_Toc11949073)

[1.6 Hardware and Software Requirement 4](#_Toc11949074)

[2.0 Class Diagram 5](#_Toc11949075)

[2.1 Data Access Layer 5](#_Toc11949076)

[2.2 ConnectionHandler.cs 6](#_Toc11949077)

[3.0 DAO for View Movie List Admin (TYUC001) 6](#_Toc11949078)

[3.1 MovieDaoSqlImpl.cs 6](#_Toc11949079)

[4.0 DAO for View Movie List Customer (TYUC002) 6](#_Toc11949080)

[4.1 MovieDaoSqlImpl.cs 7](#_Toc11949081)

[5.0 DAO for Edit Movie (TYUC003) 7](#_Toc11949082)

[5.1 MovieDaoSqlImpl.cs 7](#_Toc11949083)

[6.0 DAO for Add a Movie to Favorite (TYUC004) 8](#_Toc11949084)

[6.1 FavoriteDaoSqlImpl.cs 8](#_Toc11949085)

[7.0 DAO for View Favorite (TYUC005) 8](#_Toc11949086)

[7.1 FavoriteDaoSqlImpl.cs 8](#_Toc11949087)

[8.0 DAO for Remove Item from Favorite (TYUC006) 8](#_Toc11949088)

[8.1 FavoriteDaoSqlImpl.cs 8](#_Toc11949089)

[9.0 Standards and Guidelines 9](#_Toc11949090)

[9.1 DAO 9](#_Toc11949091)

[10.0 Submission 9](#_Toc11949092)

[10.1 Code submission instructions 9](#_Toc11949093)

[11.0 Change Log 10](#_Toc11949094)

# Introduction

## Purpose of this document

The purpose of this document is to define the ADO.NET module implementation for movieCruiser project.

## Definitions & Acronyms

|  |  |
| --- | --- |
| Definition / Acronym | Description |
| DAO | Data Access Object |
| ADO.NET | Activex Data Objects |

## Project Overview

Refer Use Case specification document for understanding the functionality and features.



## Scope

1. Creation of DAO classes and methods for reading and persisting data of moviecruiser application.

## Intended Audience

* Product Owner
* Scrum Master
* Application Architect
* Project Manager
* Test Manager
* Development Team
* Testing Team

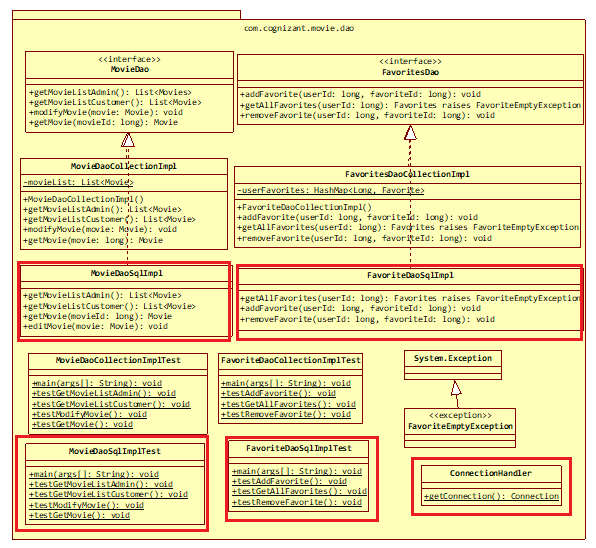
## Hardware and Software Requirement

1. Hardware Requirement:
   1. Developer PC with 4GB Ram
2. Software Requirement
   1. Git
   2. IE or Chrome
   3. .Net Framework 4.5
   4. Visual Studio Professional Edition 2015
   5. SQL Server enterprise edition 2014

# Class Diagram

## Data Access Layer

Refer the diagram below and create classes accordingly.



Dotted arrow represents implementation of an interface.

Make note that getConnection is a static method.

Test method specification is not provided in this document. Refer similar implementation in C# Specification document.

Highlighted classes are the ones that needs to be implemented in this specification.

## ConnectionHandler.cs

This class will be used by each Dao implementation class for getting the database connection.

The connection details has to be stored in a web.config file. Find the details below:

<connectionStrings>

<add name="connectionString"

connectionString="Data Source=localhost;Initial Catalog=moviecruiser;userid=root;password=password123;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

**static getConnection(): Connection**

Gets connection using ConfigurationManager Class by referring the name attribute in <connectionStrings> element present in Web.Config file

properties from previous step and return the connection.

# DAO for View Movie List Admin (TYUC001)

## MovieDaoSqlImpl.cs

**getMovieListAdmin(): List<Movie>**

1. Get connection using ConnectionHandler
2. Initialize an ArrayList of Movie
3. Using SqlCommand execute the select query that retrieves all the records from movies table
4. Iterate through the ResultSet
5. For each item in the ResultSet create a new Movie instance and add it to the ArrayList created in the step 2 and return the ArrayList

# DAO for View Movie List Customer (TYUC002)

## MovieDaoSqlImpl.cs

**getMovieListCustomer(): List<Movie>**

1. Get connection using ConnectionHandler
2. Initialize an ArrayList of Movie
3. Using SqlCommand execute the select query that retrieves the records from movie table applying the following filters:
   1. The movie is in stock and
   2. The movie is not past the expiry date
4. Iterate through the ResultSet
5. For each item in the ResultSet create a new Movie instance and add it to the ArrayList created in the step 2 and return the ArrayList

# DAO for Edit Movie (TYUC003)

## MovieDaoSqlImpl.cs

**getMovie(movieId: long): Movie**

1. Get connection using ConnectionHandler
2. Execute select query using SqlCommand that retrieves an item from movie table based on movieId.
3. Create a Movie instance and set the values for this movie instance from the first item of the ResultSet
4. Return the movie created in the previous step

**editMovie(movie: Movie): void**

1. Get connection using ConnectionHandler
2. Execute update statement using SqlCommand that modifies the values of movie table based on movieId.
3. Set the parameters of the SqlCommand and execute the statement.

# DAO for Add a Movie to Favorite (TYUC004)

## FavoriteDaoSqlImpl.cs

**addMovie(userId: long, movieId: long): void**

1. Get connection using ConnectionHandler
2. Execute insert statement using SqlCommand for inserting data into favorite table with userId and movieId.

# DAO for View Favorite (TYUC005)

## FavoriteDaoSqlImpl.cs

**getMovies(userId: long): List<Movie>**

1. Get connection using ConnectionHandler
2. Initialize an ArrayList of Movie
3. Execute update statement using SqlCommand that joins Favorite and Movie table to retrieve the list of movies associated with a specific user.
4. Iterate through the ResultSet
5. For each item in the ResultSet create a new Movie instance and add it to the ArrayList created in the step 2 and return the ArrayList

# DAO for Remove Item from Favorite (TYUC006)

## FavoriteDaoSqlImpl.cs

**removeMovie(userId: long, movieId: long): void**

1. Get connection using ConnectionHandler
2. Execute delete statement using SqlCommand for delete data into favorite table based on userId and movieId.

# Standards and Guidelines

## DAO

1. All .Net coding standards are applicable
2. Closure of connection should be done within finally block

# Submission

## Code submission instructions

Once your code is evaluated by the trainer and all the issues reported by the trainer are corrected, the code needs to be submitted to the remote repository. Follow the steps below to submit the code to remote repository.

1. In Windows Explorer go to the moviecruiser folder
2. Right click on the empty space in the right hand side of Windows Explorer and select “Git Bash here”
3. Execute the following commands

To display the added or modified files

git status

To stage the added or modified files

git add .

To display the staged files

git status

To save the code to local repository

git commit -m "jdbc"

To transfer the changes from local machine to server

git push origin master

1. Successful execution of the above commands will upload the files to the server repository.
2. Login into <https://code.cognizant.com>
3. Click on the project moviecruiser
4. Check if the files that are uploaded correctly with appropriate folder structure.

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 | Initial baseline created on 20-May-19 by Ramamoorthy Selvam | | | |
| Vx.y.z | <Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed> | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |