High-level Technical Design

HRS Application

HRS APPLICATION

High-Level Technical Design

Author: HRS Application Designer

Document Properties

Prepared by	To be reviewed by	To be approved by
Synergetics India		

Revision History

Table of Contents

Appendices:

None.

1 Introduction

The HRS Application is a web-based simulation application for the ATS Application Programming: C# Programming School. It provides basic functionalities of employee information maintenance, skills maintenance and project maintenance. This application will aid in the learning objectives of the school.

The HRS Application has four functionalities to support, specifically:

- Employee Maintenance
- Skill Category Maintenance
- Skill Detail Maintenance
- Project Maintenance

The HRS Application has identified three actors that interact with this system, the HR Manager, the Employee, and the HR Clerk. Version 1.0 of the application implements the Use Cases of the HR Manager only.

1.1 Purpose of this document

This document provides the high-level technical design for the HRS Application.

1.2 Other Related Documents

Related Documents	Short Description	Owner of Document	Link to this Document	
			Parent	Child
Not Applicable				

2 Technical Design Strategy

The application architecture approach is to divide the HRS Application into three-tier architecture, specifically; the application will be divided into Presentation Layer, Application Layer and Data Layer components.

The application will follow the Model-View-Controller (MVC) pattern. The aspx pages in the presentation layer will be the View. The aspx code behind files and corresponding controller class files will function as the Controller. Model will consist of the business layer entities.

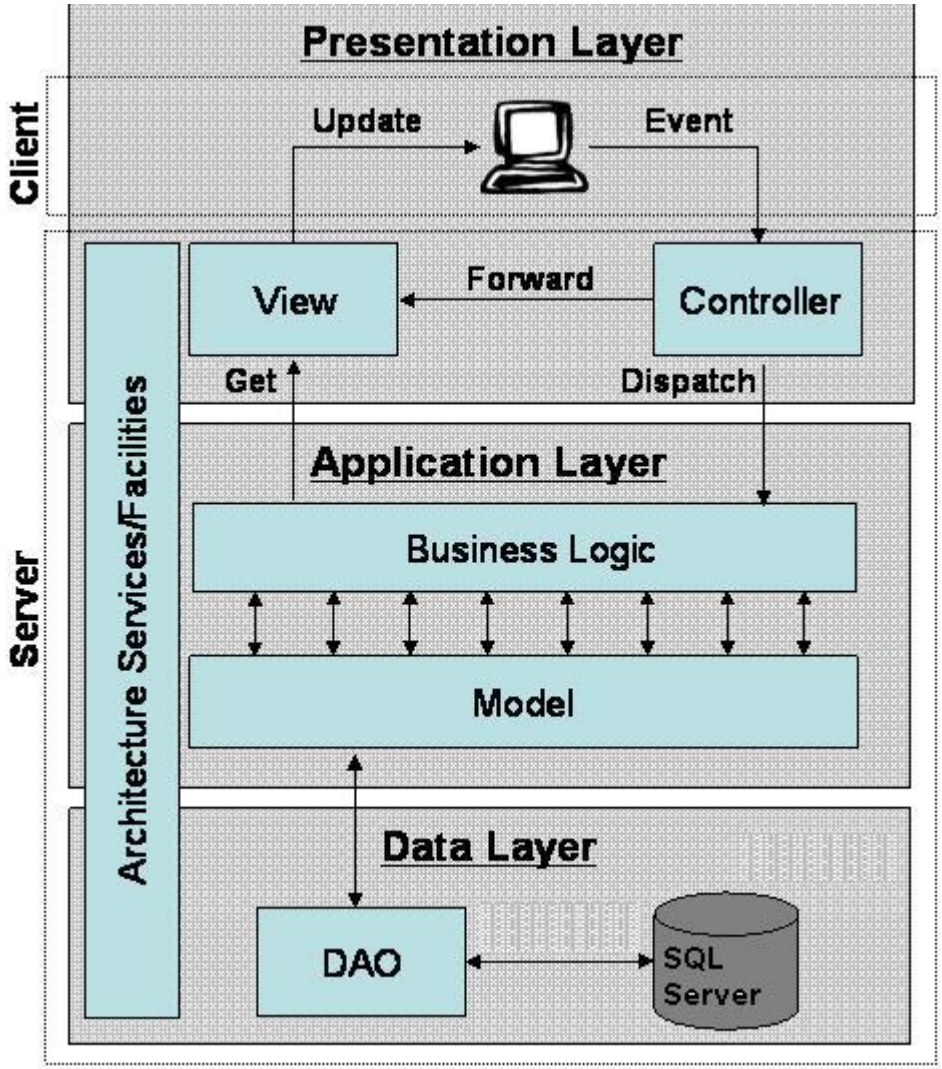

Figure 1. Application architecture diagram

Error! Not a valid link.

Figure 2. High-level Application Architecture Flow

Figure 2 illustrates the high-level application architecture flow (conversation flow) for a typical module that supports the HRS Application functionality. The following steps are followed:

1. The input to WebForm acts like a trigger to load it.
   For e.g. When user clicks on Add button, the Add WebForm is loaded.
2. The code behind of the WebForm calls the corresponding PageController class method and passes the data provided by the user its input parameters.
3. The page controller packages this data into business entity object and calls the corresponding method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the appropriate method of the corresponding Business component.
5. The Business component checks the necessary business rules and calls the appropriate method of the corresponding DAO object.
6. The DAO object makes use of the static SqlHelper class to access the SQL Server database either to fetch or insert / update the data.
7. The status or requested objects are rendered to the browser by the code-behind file.

For the ATS Application Programming: C# Programming School exercise, the concentration will be on the BCs and DAOs. This document will concentrate on the design of these artifacts.

2.1 Classes Overview

2.1.1 Presentation Layer

The Presentation tier is implemented using ASP.Net web forms and user controls. The code-behind files are in C# language. Client-side validation controls are used to avoid post-backs where ever possible.

The various pages present in this layer are listed below:

Page/Class Name	Functionality
Login.	Provides UI for login functionality
CreateEmployee.	Provides UI for create operation for Employees
UpdateEmployee.	Provides UI for edit operation for Employees
ViewEmployee.	Provides UI for view operation for Employees
SearchEmployee.	Provides UI for search operations on Employees
CreateProject.	Provides UI for create operation for Projects
UpdateProject.	Provides UI for edit operation for Projects
ViewProject.	Provides UI for view operation for Projects
SearchProject.	Provides UI for search operations on Projects
CreateSkill.	Provides UI for create operation for Skills
UpdateSkill.	Provides UI for edit operation for Skills
ViewSkill.	Provides UI for view operation for Skills
SearchSkill.	Provides UI for search operations on Skills
CreateCategory.	Provides UI for create operation for Categories
UpdateCategory.	Provides UI for edit operation for Categories
ViewCategory.	Provides UI for view operation for Categories
SearchCategory.	Provides UI for search operations on Categories

User Controls	Functionality
Menu.	Header Menu

2.1.2 Business Entities

Business Entities relate to the actual entities in the system like Employee, Project, Skill and Category.
These classes contain properties corresponding to the different columns in the corresponding database table.
For e.g.: EmployeeName, ProjectStartDate

This layer also contains custom collection classes for each entity.

The various classes present in this layer are listed below:

Class Name	Functionality
EmployeeInfo.	Provides mapping to the Employee entity.
AccentureDetail.	Provides mapping to the Employee entity.
ProjectInfo.	Provides mapping to the Project entity.
SkillInfo.	Provides mapping to the Skill entity.
CategoryInfo.	Provides mapping to the Category entity.
GenericInfoCollection.	Collection class for all the entities.

2.1.3 DataAccess

The data access layer supports all data access requirements of the Business layer and Logging classes. It includes the code used to access data. The data access layer isolates the presentation and business layers from changes happening within the data store, to a large extent.

All data access and manipulation will primarily be using Stored Procedure. Data layer has been designed specific to SQL Database.

The various classes present in this layer are listed below:

Class Name	Functionality

2.1.4 Common

This layer contains classes for Exceptions, logging and common functions

The various classes present in this layer are listed below:

Class / File Name	Functionality
Enumerators.	Contains definition of all the enumerators.
Database.	Holds the connection string property and stored procedure constants

3 Application Technical Design

3.1 Supported Use-Cases

Listed below are the Use Cases for the HRS Application.

Use Case #	Use Case Name	Use Case File Name
UC001	Maintain Employee	N.A.
UC001.1	Create Employee	UCHR01 – HR Create Employee
UC001.2	Search for Employee(s)	UCHR02 – Search Employee
UC001.3	Update Employee	UCHR03 – Update Employee
UC001.4	View Employee	N.A.
UC002	Maintain Skill Category	N.A.
UC002.1	Create Skill Category	UCHR04 – HR Create Skill Category
UC002.2	Search for Skill Category(ies)	UCHR06– HR Search Skill Category
UC002.3	Update Skill Category	UCHR05 – HR Update Skill Category

Use Case #	Use Case Name	Use Case File Name
UC002.4	View Skill Category	N.A.
UC003	Maintain Skill Detail	N.A.
UC003.1	Create Skill Detail	UCHR07 – HR Create Skill
UC003.2	Search for Skill Detail(s)	UCHR08 – HR Search Skill
UC003.3	Update Skill Detail	UCHR09 – HR Update Skill
UC003.4	View Skill Detail	N.A.
UC004	Maintain Project	N.A.
UC004.1	Create Project	UCHR10 – HR Create Project
UC004.2	Search for Project(s)	UCHR11 – HR Search Project
UC004.3	Update Project	UCHR12 – HR Update Project
UC004.4	View Project	N.A.


searchEmployees
updateEmployee
<<includes>>
viewEmployee
createEmployee
<<includes>>
<<includes>>
<<includes>>
updateSkill
Maintain Employee
<<includes>>
createSkill
<<includes>>
searchSkills
Maintain Skill
<<includes>>
viewSkill
HR Actor
Maintain SkillCategory
<<includes>>
updateCategory
Maintain Project
<<includes>>
createCategory
<<includes>>
viewCategory
<<includes>>
searchCategories
searchProjects
<<includes>>
<<includes>>
createProject
<<includes>>
updateProject
<<includes>>
viewProject

3.2 Sequence Diagrams

This section discusses the supported activities of the application. The diagram shows the interaction of related classes to implement the corresponding use-case.

No	Sequence Name	Use-Case Name
4.2.1	Create Employee	Create Employee
4.2.2	Update Employee	Update Employee
4.2.3	View Employee	View Employee
4.2.4	Search Employees	Search for Employee(s)
4.2.5	Create Project	Create Project
4.2.6	Update Project	Update Project
4.2.7	View Project	View Project
4.2.8	Search Projects	Search for Project(s)
4.2.9	Create Skill	Create Skill Detail
4.2.10	Update Skill	Update Skill Detail
4.2.11	View Skill	View Skill Detail

No	Sequence Name	Use-Case Name
4.2.12	Search Skills	Search for Skill Detail(s)
4.2.13	Create SkillCategory	Create Skill Category
4.2.14	Update SkillCategory	Update Skill Category
4.2.15	View SkillCategory	View Skill Category
4.2.16	Search SkillCategories	Search for Skill Category(ies)

3.2.1 Create Employee Activity

Error! Not a valid link.

This diagram describes the high-level overview of the Create Employee Sequence flow.

1. When user clicks on Add Employee, the Create Employee page is loaded.
2. The code behind class of Create Employee form calls the CreateEmployee method of the EmployeeController class and passes the data provided by the user as its input parameters.
3. The EmployeeController class packages this data into an EmployeeInfo business entity object and calls the CreateEmployee method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the CreateEmployee method of the EmployeeBC business component.
5. The EmployeeBC checks the necessary business rules and calls the CreateEmployee method of the EmployeeDAO object.
6. The EmployeeDAO object makes use of the static SqlHelper class to access the SQL Server database to insert the employee details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to EmployeeDAO.
8. EmployeeDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Create Employee class

3.2.2 Update Employee Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Update Employee Sequence flow.

1. When user clicks on Update Employee, the Update Employee page is loaded.
2. The code behind class of Update Employee form calls the UpdateEmployee method of the EmployeeController class and passes the data provided by the user as its input parameters.
3. The EmployeeController class packages this data into an EmployeeInfo business entity object and calls the UpdateEmployee method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the UpdateEmployee method of the EmployeeBC business component.
5. The EmployeeBC checks the necessary business rules and calls the UpdateEmployee method of the EmployeeDAO object.
6. The EmployeeDAO object makes use of the static SqlHelper class to access the SQL Server database to update the employee details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to EmployeeDAO.
8. EmployeeDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Update Employee class

3.2.3 View Employee Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the View Employee Sequence flow.

1. When user clicks on View Employee, the View Employee page is loaded.
2. The code behind class of View Employee form calls the SearchEmployee method of the EmployeeController class and passes the employee id provided by the user as its input parameter.
3. The EmployeeController class calls the SearchEmployee method of the HRManager class which acts as a Façade for the Business Layer. The employee id is passed as the input parameter.
4. The HRManager invokes the SearchEmployee method of the EmployeeBC business component.
5. The EmployeeBC calls the FindByPK method of the EmployeeDAO object.

6. The EmployeeDAO object makes use of the static SqlHelper class to access the SQL Server database and get the employee details using the ExecuteDataSet method.
7. SqlHelper returns the employee details in the form of a dataset to EmployeeDAO.
8. EmployeeDAO returns the employee details in the form of a datatable to EmployeeBC
9. EmployeeBC returns this datatable to HRManager which passes it on to EmployeeController.
10. The EmployeeController class packages this data into an EmployeeInfo business entity object and returns it to View Employee class
11. View Employee class reads the EmployeeInfo properties and displays them on the browser.

3.2.4 Search Employees Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Search Employee Sequence flow.

1. When user clicks on Search, the Search Employee page is loaded.
2. The code behind class of Search Employee form calls the SearchEmployees method of the EmployeeController class and passes the employee first name and lastname provided by the user as its input parameter.
3. The EmployeeController class calls the SearchEmployees method of the HRManager class which acts as a Façade for the Business Layer. The employee's First Name and Last Name are passed as the input parameters.
4. The HRManager invokes the SearchEmployees method of the EmployeeBC business component.
5. The EmployeeBC calls the FindByName method of the corresponding DAO object.
6. The EmployeeDAO object makes use of the static SqlHelper class to access the SQL Server database and get the details of all employees matching the search criteria using the ExecuteDataSet method.
7. SqlHelper returns the employees in the form of a dataset to EmployeeDAO.
8. EmployeeDAO returns the employees in the form of a datatable to EmployeeBC
9. EmployeeBC returns this datatable to HRManager which passes it on to EmployeeController.
10. The EmployeeController class packages this data into a collection of EmployeeInfo business entity objects and returns it to Search Employee class
11. Search Employee class reads the EmployeeInfo properties and displays them on the browser.

3.2.5 Create Project Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Create Project Sequence flow.

1. When user clicks on Add Project, the Create Project page is loaded.
2. The code behind class of Create Project form calls the CreateProject method of the ProjectController class and passes the data provided by the user as its input parameters.
3. The ProjectController class packages this data into an ProjectInfo business entity object and calls the CreateProject method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the CreateProject method of the ProjectBC business component.
5. The ProjectBC checks the necessary business rules and calls the CreateProject method of the ProjectDAO object.
6. The ProjectDAO object makes use of the static SqlHelper class to access the SQL Server database to insert the Project details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to ProjectDAO.
8. ProjectDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Create Project class

3.2.6 Update Project Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Update Project Sequence flow.

1. When user clicks on Update Project, the Update Project page is loaded.
2. The code behind class of Update Project form calls the UpdateProject method of the ProjectController class and passes the data provided by the user as its input parameters.
3. The ProjectController class packages this data into an ProjectInfo business entity object and calls the UpdateProject method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the UpdateProject method of the ProjectBC business component.
5. The ProjectBC checks the necessary business rules and calls the UpdateProject method of the ProjectDAO object.
6. The ProjectDAO object makes use of the static SqlHelper class to access the SQL Server database to update the Project details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to ProjectDAO.
8. ProjectDAO returns a Boolean value based on the number of records affected.

9. This bool value is propagated back to the Update Project class

3.2.7 View Project Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the View Project Sequence flow.

1. When user clicks on View Project, the View Project page is loaded.
2. The code behind class of View Project form calls the SearchProject method of the ProjectController class and passes the Project id provided by the user as its input parameter.
3. The ProjectController class calls the SearchProject method of the HRManager class which acts as a Façade for the Business Layer. The Project id is passed as the input parameter.
4. The HRManager invokes the SearchProject method of the ProjectBC business component.
5. The ProjectBC calls the FindByPK method of the ProjectDAO object.
6. The ProjectDAO object makes use of the static SqlHelper class to access the SQL Server database and get the Project details using the ExecuteDataSet method.
7. SqlHelper returns the Project details in the form of a dataset to ProjectDAO.
8. ProjectDAO returns the Project details in the form of a datatable to ProjectBC
9. ProjectBC returns this datatable to HRManager which passes it on to ProjectController.
10. The ProjectController class packages this data into an ProjectInfo business entity object and returns it to View Project class
11. View Project class reads the ProjectInfo properties and displays them on the browser.

3.2.8 Search Projects Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Search Project Sequence flow.

1. When user clicks on Search, the Search Project page is loaded.
2. The code behind class of Search Project form calls the SearchProjects method of the ProjectController class and passes the Project name provided by the user as its input parameter.

3. The ProjectController class calls the SearchProjects method of the HRManager class which acts as a Façade for the Business Layer. The Project's Name is passed as the input parameter.
4. The HRManager invokes the SearchProjects method of the ProjectBC business component.
5. The ProjectBC calls the FindByName method of the corresponding DAO object.
6. The ProjectDAO object makes use of the static SqlHelper class to access the SQL Server database and get the details of all Projects matching the search criteria using the ExecuteDataSet method.
7. SqlHelper returns the Projects in the form of a dataset to ProjectDAO.
8. ProjectDAO returns the Projects in the form of a datatable to ProjectBC
9. ProjectBC returns this datatable to HRManager which passes it on to ProjectController.
10. The ProjectController class packages this data into a collection of ProjectInfo business entity objects and returns it to Search Project class
11. Search Project class reads the ProjectInfo properties and displays them on the browser.

3.2.9 Create Skill Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Create Skill Sequence flow.

1. When user clicks on Add Skill, the Create Skill page is loaded.
2. The code behind class of Create Skill form calls the CreateSkill method of the SkillController class and passes the data provided by the user as its input parameters.
3. The SkillController class packages this data into an SkillInfo business entity object and calls the CreateSkill method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the CreateSkill method of the SkillBC business component.
5. The SkillBC checks the necessary business rules and calls the CreateSkill method of the SkillDAO object.
6. The SkillDAO object makes use of the static SqlHelper class to access the SQL Server database to insert the Skill details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to SkillDAO.
8. SkillDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Create Skill class

3.2.10 Update Skill Sequence

Error! Not a valid link.

This diagram describes the high-level overview of the Update Skill Sequence flow.

1. When user clicks on Update Skill, the Update Skill page is loaded.
2. The code behind class of Update Skill form calls the UpdateSkill method of the SkillController class and passes the data provided by the user as its input parameters.
3. The SkillController class packages this data into an SkillInfo business entity object and calls the UpdateSkill method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the UpdateSkill method of the SkillBC business component.
5. The SkillBC checks the necessary business rules and calls the UpdateSkill method of the SkillDAO object.
6. The SkillDAO object makes use of the static SqlHelper class to access the SQL Server database to update the Skill details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to SkillDAO.
8. SkillDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Update Skill class

3.2.11 View Skill Sequence

Error! Not a valid link.

This diagram describes the sequential flow for displaying skill detail.

1. When user clicks on View Skill, the View Skill page is loaded.
2. The code behind class of View Skill form calls the SearchSkill method of the SkillController class and passes the Skill id provided by the user as its input parameter.
3. The SkillController class calls the SearchSkill method of the HRManager class which acts as a Façade for the Business Layer. The Skill id is passed as the input parameter.
4. The HRManager invokes the SearchSkill method of the SkillBC business component.
5. The SkillBC calls the FindByPK method of the SkillDAO object.

6. The SkillDAO object makes use of the static SqlHelper class to access the SQL Server database and get the Skill details using the ExecuteDataSet method.
7. SqlHelper returns the Skill details in the form of a dataset to SkillDAO.
8. SkillDAO returns the Skill details in the form of a datatable to SkillBC
9. SkillBC returns this datatable to HRManager which passes it on to SkillController.
10. The SkillController class packages this data into an SkillInfo business entity object and returns it to View Skill class
11. View Skill class reads the SkillInfo properties and displays them on the browser.

3.2.12 Search Skills Sequence

Error! Not a valid link.

This diagram describes the sequential flow for searching skills.

1. When user clicks on Search, the Search Skill page is loaded.
2. The code behind class of Search Skill form calls the SearchSkills method of the SkillController class and passes the Skill name provided by the user as its input parameter.
3. The SkillController class calls the SearchSkills method of the HRManager class which acts as a Façade for the Business Layer. The Skill's Name is passed as the input parameter.
4. The HRManager invokes the SearchSkills method of the SkillBC business component.
5. The SkillBC calls the FindByName method of the corresponding DAO object.
6. The SkillDAO object makes use of the static SqlHelper class to access the SQL Server database and get the details of all Skills matching the search criteria using the ExecuteDataSet method.
7. SqlHelper returns the Skills in the form of a dataset to SkillDAO.
8. SkillDAO returns the Skills in the form of a datatable to SkillBC
9. SkillBC returns this datatable to HRManager which passes it on to SkillController.
10. The SkillController class packages this data into a collection of SkillInfo business entity objects and returns it to Search Skill class
11. Search Skill class reads the SkillInfo properties and displays them on the browser.

3.2.13 Create Category Sequence

Error! Not a valid link.

This diagram describes the sequential flow for creating a category.

1. When user clicks on Add Category, the Create Category page is loaded.
2. The code behind class of Create Category form calls the CreateCategory method of the CategoryController class and passes the data provided by the user as its input parameters.
3. The CategoryController class packages this data into an CategoryInfo business entity object and calls the CreateCategory method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the CreateCategory method of the CategoryBC business component.
5. The CategoryBC checks the necessary business rules and calls the CreateCategory method of the CategoryDAO object.
6. The CategoryDAO object makes use of the static SqlHelper class to access the SQL Server database to insert the Category details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to CategoryDAO.
8. CategoryDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Create Category class

3.2.14 Update Category Sequence

Error! Not a valid link.

This diagram describes the sequential flow for updating a category.

1. When user clicks on Update Category, the Update Category page is loaded.
2. The code behind class of Update Category form calls the UpdateCategory method of the CategoryController class and passes the data provided by the user as its input parameters.
3. The CategoryController class packages this data into an CategoryInfo business entity object and calls the UpdateCategory method of the HRManager class which acts as a Façade for the Business Layer. The business entity object is passed as the input parameter.
4. The HRManager invokes the UpdateCategory method of the CategoryBC business component.
5. The CategoryBC checks the necessary business rules and calls the UpdateCategory method of the CategoryDAO object.
6. The CategoryDAO object makes use of the static SqlHelper class to access the SQL Server database to update the Category details using the ExecuteNonQuery method.
7. SqlHelper returns the number of records affected to CategoryDAO.
8. CategoryDAO returns a Boolean value based on the number of records affected.
9. This bool value is propagated back to the Update Category class

3.2.15 View Category Sequence

Error! Not a valid link.

This diagram describes the sequential flow for displaying category detail.

1. When user clicks on View Category, the View Category page is loaded.
2. The code behind class of View Category form calls the SearchCategory method of the CategoryController class and passes the Category id provided by the user as its input parameter.
3. The CategoryController class calls the SearchCategory method of the HRManager class which acts as a Façade for the Business Layer. The Category id is passed as the input parameter.
4. The HRManager invokes the SearchCategory method of the CategoryBC business component.
5. The CategoryBC calls the FindByPK method of the CategoryDAO object.
6. The CategoryDAO object makes use of the static SqlHelper class to access the SQL Server database and get the Category details using the ExecuteDataSet method.
7. SqlHelper returns the Category details in the form of a dataset to CategoryDAO.
8. CategoryDAO returns the Category details in the form of a datatable to CategoryBC
9. CategoryBC returns this datatable to HRManager which passes it on to CategoryController.
10. The CategoryController class packages this data into an CategoryInfo business entity object and returns it to View Category class
11. View Category class reads the CategoryInfo properties and displays them on the browser.

3.2.16 Search Categories Sequence

Error! Not a valid link.

This diagram describes the sequential flow for searching categories.

1. When user clicks on Search, the Search Category page is loaded.
2. The code behind class of Search Category form calls the SearchCategories method of the CategoryController class and passes the Category name provided by the user as its input parameter.
3. The CategoryController class calls the SearchCategories method of the HRManager class which acts as a Façade for the Business Layer. The Category's Name is passed as the input parameter.

4. The HRManager invokes the SearchCategories method of the CategoryBC business component.
5. The CategoryBC calls the FindByName method of the corresponding DAO object.
6. The CategoryDAO object makes use of the static SqlHelper class to access the SQL Server database and get the details of all Categories matching the search criteria using the ExecuteDataSet method.
7. SqlHelper returns the Categories in the form of a dataset to CategoryDAO.
8. CategoryDAO returns the Categories in the form of a datatable to CategoryBC
9. CategoryBC returns this datatable to HRManager which passes it on to CategoryController.
10. The CategoryController class packages this data into a collection of CategoryInfo business entity objects and returns it to Search Category class
11. Search Category class reads the CategoryInfo properties and displays them on the browser.

4 Design Issues

The following table outlines all known open and closed (those with a date resolved entry) issues/risks identified with this technical class specification.

Issue/Risk Description	Resolution/Mitigation	Date Raised	Responsible	Date Resolved
1. The current implementation of the class only supports the HR Manager actor. The support for the other identified actors has not been taken into consideration with this document.	9. This is a specific implementation decision and the mitigation implemented is to allow only HR Manager users to access the system	9 March 2006.	Application Architect	