ARCHITECTURE DESIGN DOCUMENT

CAPABILITY CONTROL LIST

(Submitted in partial fulfillment of the requirements of the degree of Master Software Engineering)

SRUNOKSHI KANIYUR PREMA NEELAKANTAN

CIS 895 – MSE PROJECT

KANSAS STATE UNIVERSITY

TABLE OF CONTENT

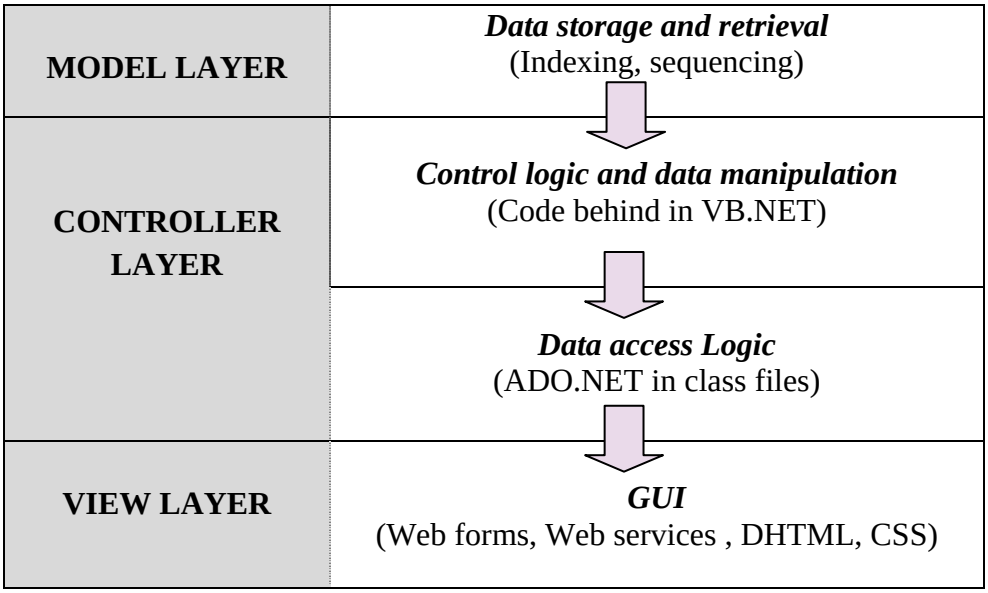
1. INTRODUCTION

The main purpose of this document is to provide the architecture design for 'Capability control list' application. This application follows the Model View Controller (MVC) architecture pattern i.e. it contains a Model (i.e. Data tier), View (i.e. a Presentation tier) and Controller (i.e. a business-logic tier).

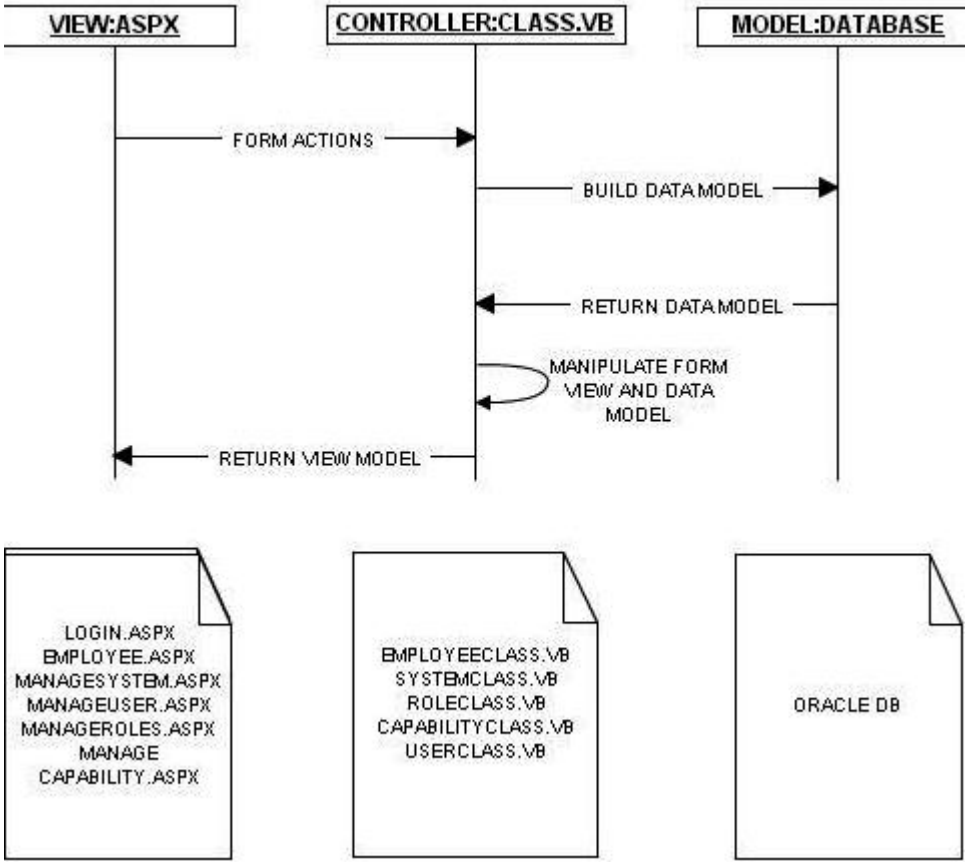
2. ARCHITECTURE TIER AND TECHNOLOGY STACK

This application follows the MVC architecture and an explanation for each layer is given. The main advantage in using this architecture is,

- The 'Model' and the 'Controller' layers are separated to enhance the readability for the developer who can later update the code easily
- The logic used in the 'Controller' can be reused across other component/application
- Traffic between 'Controller' and 'Model' layers can be reduced to enhance the security of the data.



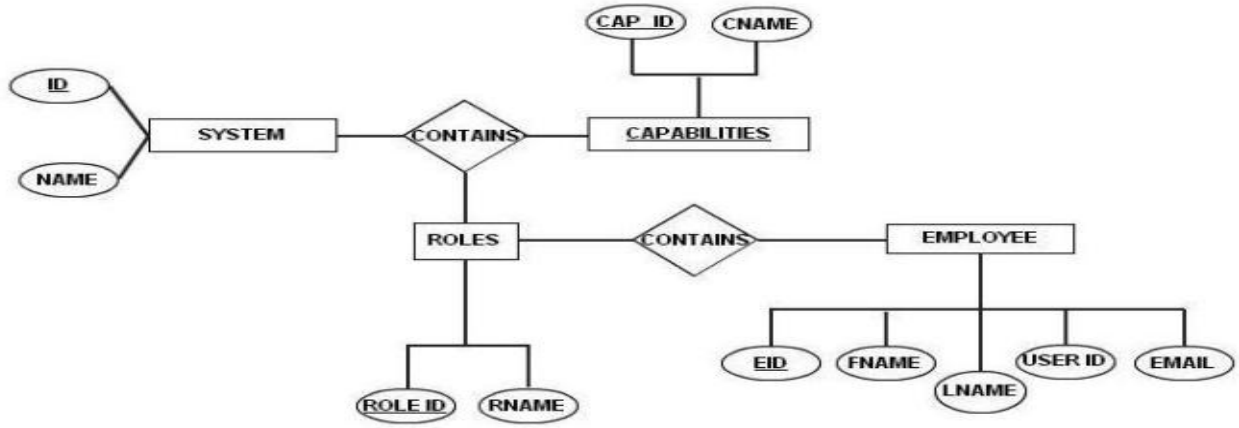
3. EXPLANATION AND USE OF MODEL VIEW CONTROLLER DESIGN PATTERN (MVC)



MODEL LAYER

This layer is responsible for storing and retrieving data in the database tables which the application uses. This layer is independent of the view or controller and so the data in these tables can be used across any number of application views. Oracle is the database server for this application. The model responds to the request made by the controller and displays the updated data in the view layer. Data in the table is encrypted for security purposes.

Entity-relationship diagram for Capability Control List (CCL)



VIEW LAYER

This layer is helpful in presenting the application to the end users such that the request and the corresponding processed results are understandable to the user. The view layer for the Capability control list is used for designing the web forms using web controls like textbox, grid view etc. and web services.

The following are some of the web forms that are used in this application by '**administrators**'.

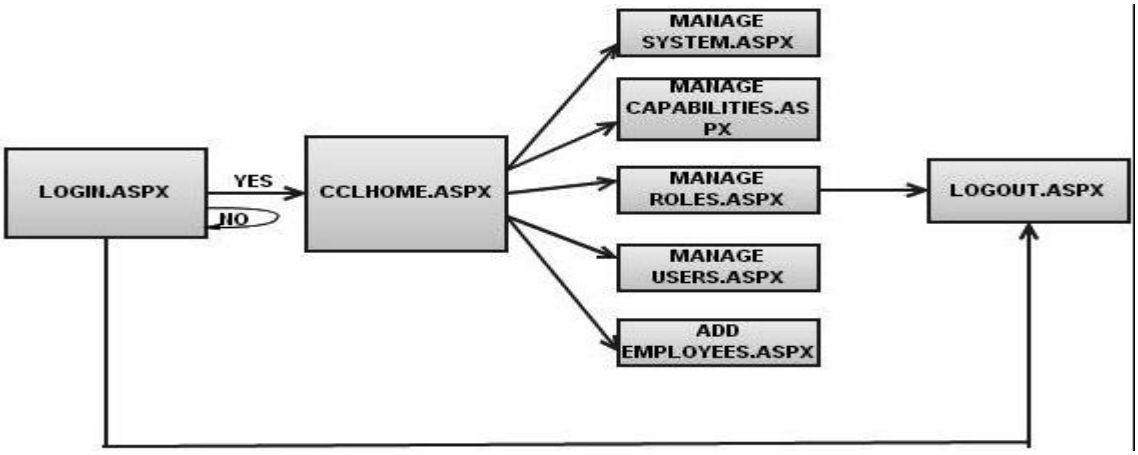
ASP.NET FORMS	**EXPLANATION**
CCLHome.aspx	This is the home for administrators
ManageSystem.aspx	This is used for adding the application which capabilities can be controlled by CCL application
ManageUser.aspx	This is used for managing users across different application
ManageRoles.aspx	This is used for managing roles for the systems controlled by CCL
ManageCapability.aspx	This is used for managing capabilities for the systems controlled by CCL
Login.aspx	This is the login page of the application to authenticate users

The following are some of the web pages that are used in this application by '**client users**'.

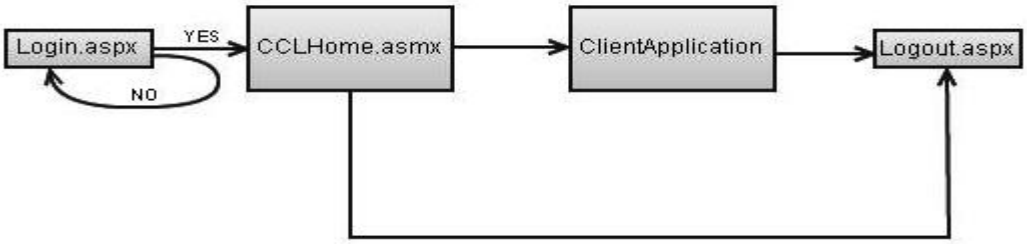
ASP.NET FORMS	**EXPLANATION**
CCLWSHome.asmx	This is the home page for client users
Login.aspx	This is the login page of the application to authenticate users

Page flow diagram for Capability Control List (CCL)

Administrators



Client users

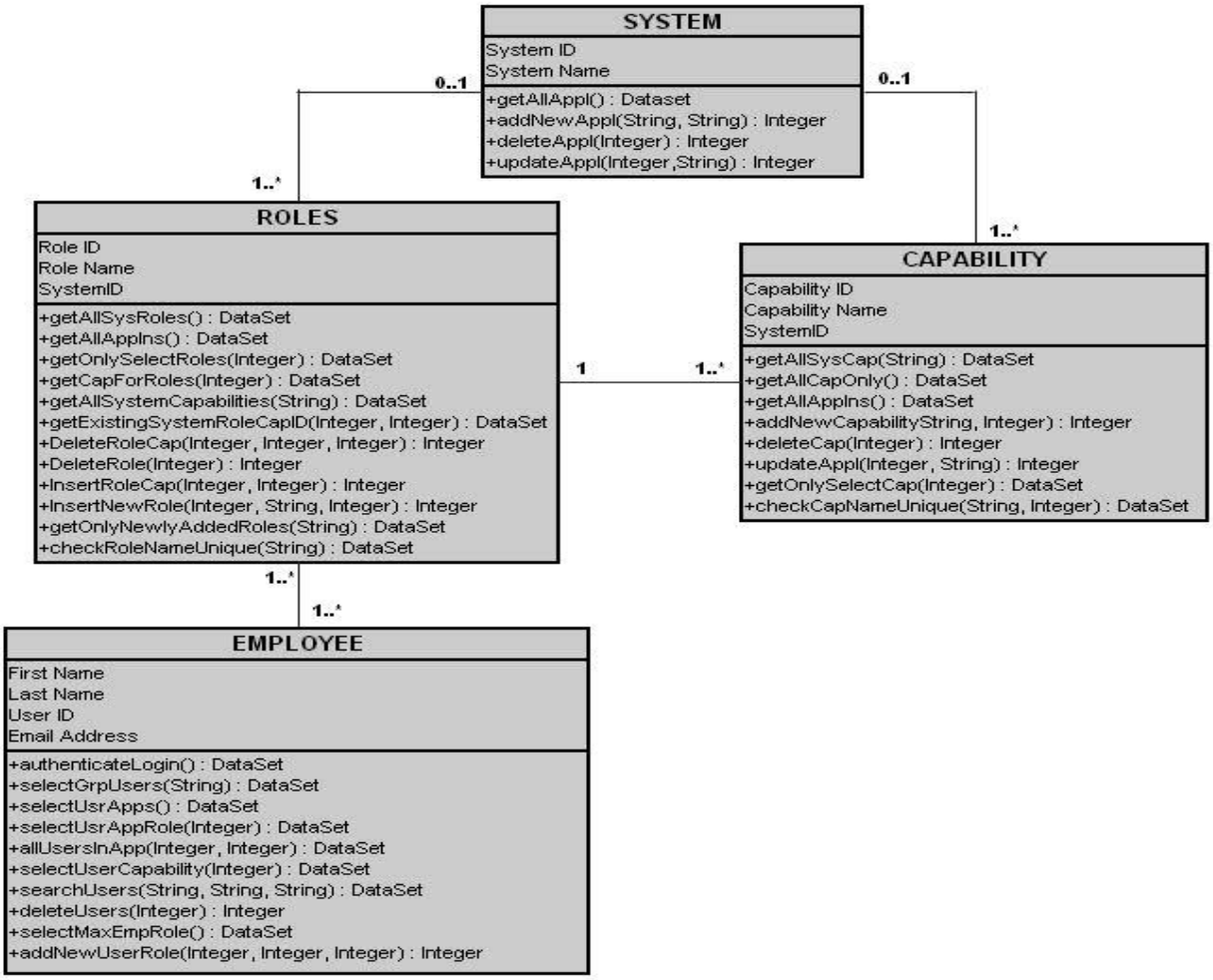


BUSINESS-LOGIC LAYER

Control logic and data manipulation - This layer is responsible for handling events that are triggered by the web forms and performs evaluations and calculations

Data access Logic – This layer is responsible for moving the data between the 'Model' and 'View' tier

Class diagram for Capability Control List (CCL),



The diagram above lists all the classes that are used to build the CCL application. There are four important classes for this application i.e. System, Roles, Capability, and Employee. Each of the four classes has its own property, functions and associations. System class is used for maintaining all the applications that are controlled by CCL. Roles class is used for creating the roles based on responsibilities. Capabilities class is used for managing the capabilities used in an application and role. Employee class is for maintaining the users working on CCL and other client application

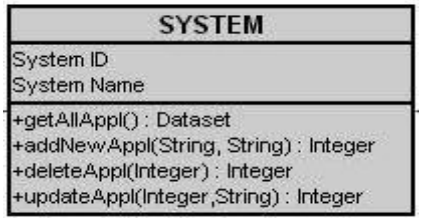
Associations between classes,

- Each system is related with atleast one (i.e. one or more) role. Each role is associated with atmost one (i.e. zero or one) system
- Each system is associated with atleast one (i.e. one or more) role capabilities. Each capability is associated with atmost one (i.e. zero or one) system.
- Each role is associated with atleast one (i.e. one or more) role capabilities.

- Each role is associated with atleast one (i.e. one or more) role users

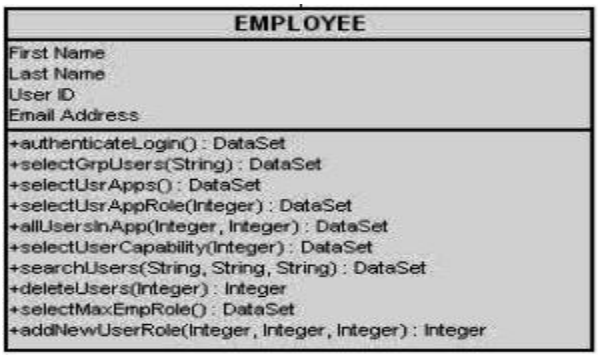
Class Descriptions

SYSTEM



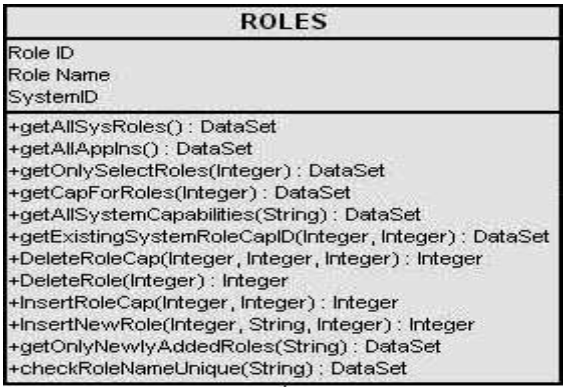
This class is useful in maintaining the applications that are controlled by Capability Control List (CCL). The main objective of CCL is to control the capabilities of the users working in an application and such applications are added to CCL using addNewAppl() method. Application name can be updated and the application is deleted if it is no longer controlled by CCL.

USER



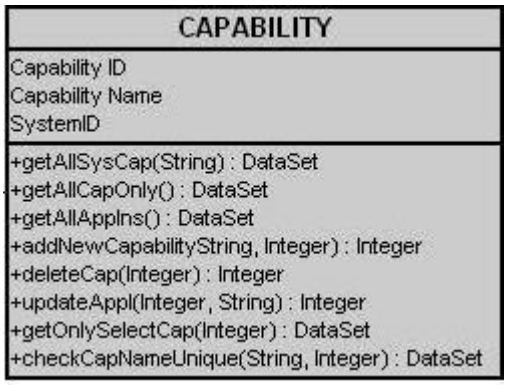
This class deals with all the users working on CCL and other client applications controlled by CCL. Users whose identity is verified during the login using authenticateLogin() method are successfully given permission to access the corresponding applications. This class contains four attributes, first name, last name, user id, and email. Users can be searched using the searchUsers() method and those users are added to corresponding roles. If the user responsibility changes such users are deleted from their corresponding roles and this operation is performed by deleteUsers method. They are later added to their new roles using addNewUserRole() method.

ROLES



This class is essential in maintaining the roles for an application. Each application can have 'n' of roles. The roles are classified depending on the responsibilities. Each role can have 'n' of users and also capabilities. These capabilities must be associated with the application of that role. New roles can be added to an application using InsertNewRole() method. The associated capabilities are added using InsertRoleCap() method. Role name uniqueness for that application is achieved by checkRoleNameUnique() method.

CAPABILITIES



This class is essential in maintaining the capabilities for an application. Capabilities are also used for roles and the user's access permission is controlled by these capabilities. New capabilities are added to the application using addNewCapability() method.

4. REFERENCES

http://en.wikipedia.org/wiki/Multitier_architecture

http://en.wikipedia.org/wiki/File:Overview_of_a_three-tier_application_vectorVersion.svg

http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller#.NET