Printshop Workflow Automation System
P.W.A.S

Deliverable #2 – Part II
Object Design Document

Presented by:

Dulcardo Arteaga • Naveen Gowda • Larissa Guerrero
Erik Kessler • Lenny Markus • Javier Mesa • Rolando Vicaria

1. Introduction

The goal for PWAS (Printshop Workflow Automation System) is to make production more efficient for XYZ Printing Co. After careful analysis of the requirements, it was determined that the optimum design scheme for this system would be a three tiered architecture. The logic layer will be accessed through a thin client, in this case it will be a browser based client. This will be the only interface to the system and will be used both by XYZ's employees as well as their customers.

The system must be designed in such a way that it can be easily scaled to manage a large number of users while maintaining adequate response times to user requests (less than 30 seconds). The logic & storage layer should be designed and implemented in such a way that they are not strongly coupled, and can be physically apart from each other. The logic layer will be implemented using C#/ISS/ASP technologies, and the storage layer will be implemented using SQLServer.

Regarding the interface, the GUI design team must take into account the fact that the system will mark a departure from long standing traditions at XYZ, so the new interface should have a very small learning curve. All functionalities should be very straightforward and clear for users, and it must be kept in mind that the main purpose of the system is to increase efficiency and minimize human caused errors. Therefore, the interface's goal will be to make the system conducive to a smooth, trouble-free workflow that will be powerful, yet simple to grasp for all users. The interface must be consistent and fully functional on Firefox 3.5+, and IE7+. Support for other browsers is not required at this time.

The subsystem decomposition from which the systems and objects in this document were developed is summarized in the figure below:

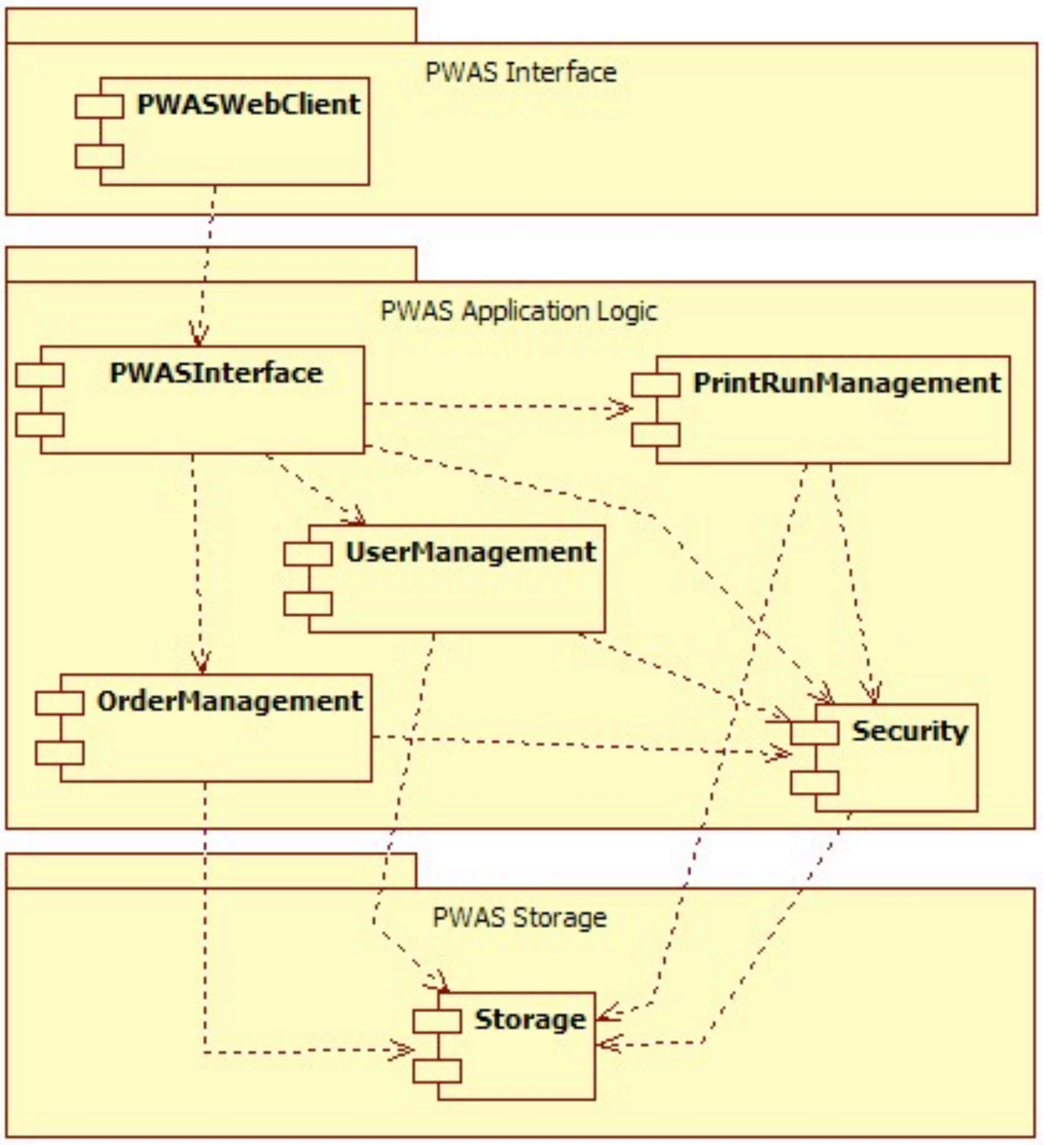


Figure 1.1 - Subsystem Decomposition Overview

1.1 Object design trade-offs

The decision to use C#/ISS/ASP technologies was made based on the fact that the LINQ (Link to integrated Query) functionality that they provide will allow for a very short development phase, as the majority of the heavy coding groundwork will be created automatically. Because of this, logic layer work will be greatly simplified and only limited to implementing the basic class functionalities that are not created automatically, however, no additional libraries/packages/frameworks will be acquired for this project, as there is no commercial or free product currently on the market that would enable production to proceed faster. All remaining code must be created from scratch.

Logic and Storage will be created in such a way that they are very loosely coupled. Even though this will require additional programming hours, the ultimate goal is to allow the system to be very scalable and flexible. Initially, both the Storage and Logic layers will be hosted physically in the same machine, but as the system grows, hardware will need to be upgraded to cope with the increased stress. At this point, our design scheme will pay off, since storage can be trivially relocated to a faster machine with minimal disruption to the system.

Interface development has been limited to only two browsers (IE7+, Firefox 3.5+) as these are currently the two dominant browsers. Testing and debugging for a wide range of browsers would be cost and time prohibitive; especially when taking into account that any gains would be minimal at best. Interface development must adhere to HTML4, XHTML 1.0 and DOM2 HTML Standards, so it is reasonable to expect that the interface will still work with most modern browsers, with minimal cosmetic differences. Only the aforementioned browsers must be thoroughly tested.

1.2 Interface documentation guidelines

Given that .NET Framework technologies will be used throughout this project, the .NET design guidelines will be followed for all aspects of the system. This will insure that the code will be easy to understand and maintain using available .NET Framework tools. For the complete reference please see section 1.4

1.3 Definitions, acronyms, and abbreviations

- PWAS: Printshop Workflow Automation System. Name of the system.

1.4 References

.NET Framework Design Guidelines: http://msdn.microsoft.com/en-us/library/ms229042.aspx

2. Packages

User Management Package

Filename	Classes contained
ManageUsers.cs	ManageUsers
Users.cs	Users
User.cs	User

Overview

The User Management Package consists of ManageUsers, Users, and User classes. This package encapsulates all the objects necessary to manage a user account. The ManageUsers class is mainly a controller class for isolating all the logic pertaining to Users. It interacts with the Users class to retrieve records from the database. The Users class is a utility class for abstracting access to the database. To this end, a single record from the database is further abstracted by the User class.

Dependencies with other packages

The User Management Package depends primarily on the Security Management Package to assert an individual's permission to access information from the User class. This is done by assigning users a role which has permissions defined for all members of that role. A user is also composed of a list of Order objects that the user owns.

Usage

Interaction with the User Management Package will be through the ManageUsers class. In response to user input, the web pages that compose the UI of the system will call on methods of the ManageUsers class to view/edit/create/delete User records.

Order Management Package

Filename	Classes contained
ManageOrders.cs	ManageOrders
Orders.cs	Orders
Order.cs	Order

Overview

The Order Management Package consists of the ManageOrders, Orders, and Order classes. This package encapsulates all the objects necessary to manage an order. The ManageOrders class is mainly a controller class for isolating all the logic pertaining to Orders. It interacts with the Orders class to retrieve records from the database. The Orders class is a utility class for abstracting access to the database. To this end, a single record from the database is further abstracted by the Order class.

Dependencies with other packages

The Order Management Package depends primarily on the Security Management Package to assert an individual's permission to access information from the Order class. An Order keeps a reference to its owner User object from the User Management Package.

Usage

Interaction with the Order Management Package will be through the ManageOrders class. In response to user input, the web pages that compose the UI of the system will call on methods of the ManageOrders class to view/edit/create/delete Order records.

<u>PrintRun Management Package</u>

Filename	Classes contained
ManagePrintRuns.cs	ManagePrintRuns
PrintRuns.cs	PrintRuns
PrintRun.cs	PrintRun

Overview

The PrintRun Management Package consists of the ManagePrintRuns, PrintRuns, and PrintRun classes. This package encapsulates all the objects necessary to manage an order. The ManagePrintRuns class is mainly a controller class for isolating all the logic pertaining to PrintRuns. It interacts with the PrintRuns class to retrieve records from the database. The PrintRuns class is a utility class for abstracting access to the database. To this end, a single record from the database is further abstracted by the PrintRun class.

Dependencies with other packages

The PrintRun Management Package depends primarily on the Security Management Package to assert an individual's permission to access information from the PrintRun class. A PrintRun is composed of a list of Order objects from the Order Management Package.

Usage

Interaction with the Printrun Management Package will be through the ManagePrintRuns class. In response to user input, the web pages that compose the UI of the system will call on methods of the ManagePrintRuns class to view/edit/create/delete PrintRun records.

Security Management Package

Filename	Classes contained
ManageSecurity.cs	ManageSecurity
Roles.cs	Roles
Role.cs	Role
RolePermissions.cs	RolePermissions
RolePermission.cs	RolePermission

Overview

The Security Management Package consists of the ManageSecurity, Roles, Role, RolePermissions, and RolePermission classes. This package encapsulates all the objects necessary to manage user authorization throughout the application. The ManageSecurity class is mainly a controller class for isolating all the logic pertaining to Roles and RolePermissions. It interacts with the RolePermissions and Roles class to retrieve records from the database. The Roles and RolePermissions classes are utility classes for abstracting access to the database. To this end, single records from the database are further abstracted by the Role and RolePermission classes.

Dependencies with other packages

The Security Management Package does not have any dependencies to other packages. However, all other packages depend on it.

Usage

Interaction with the Security Management Package will be through the ManageSecurity class. All webpages will access the ManageSecurity class on page load in order to verify which actions the user is authorized to perform and render the appropriate controls accordingly. Furthermore, it will be used to verify that users are not trying to access resources outside of the scope of their authorization. Administrators will have the ability to define the RolePermissions for each Role and then specify the Role for each user.

3. Class interfaces

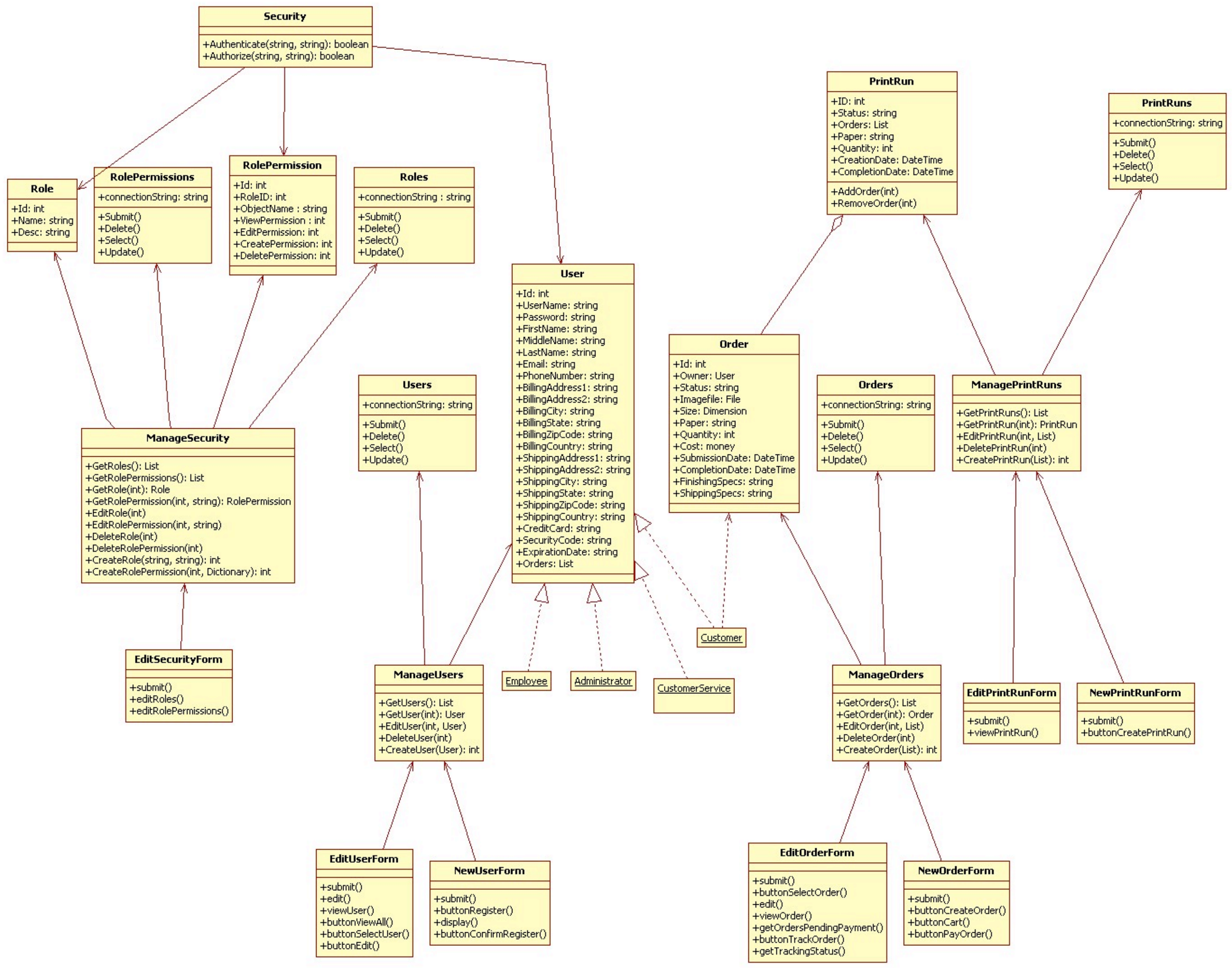
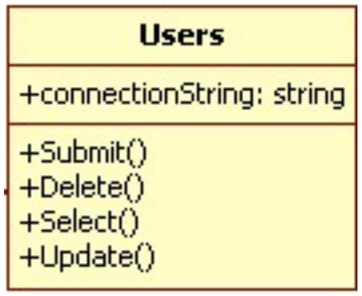


Figure 1.2 – Main Object Diagram for PWAS

Users Class

The Users class represents the collection of User objects that have been entered into the system. It allows access to the aggregate of the User objects, useful for pages that display all users and allow the user to select one for further inspection or editting.



Operations

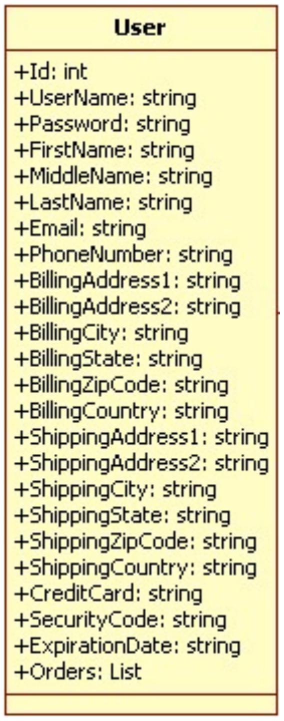
- Submit()
  Submit() is a public function which does not return any value. It is used to commit changes to User in the database.
- Delete()
  Delete () is a public function which does not return any value. It is used to delete an User in the database.
- Select()
  Select () is a public function which does not return any value. It is used to select an User in the database.
- Update()
  Update () is a public function which does not return any value. It is used to Update changes to an User in the database.

Dependencies

No dependencies.

User Class

The User class represents a single user of the system, and holds all information pertaining to that particular person. A User can be many types, such as an administrator, employee, or customer.

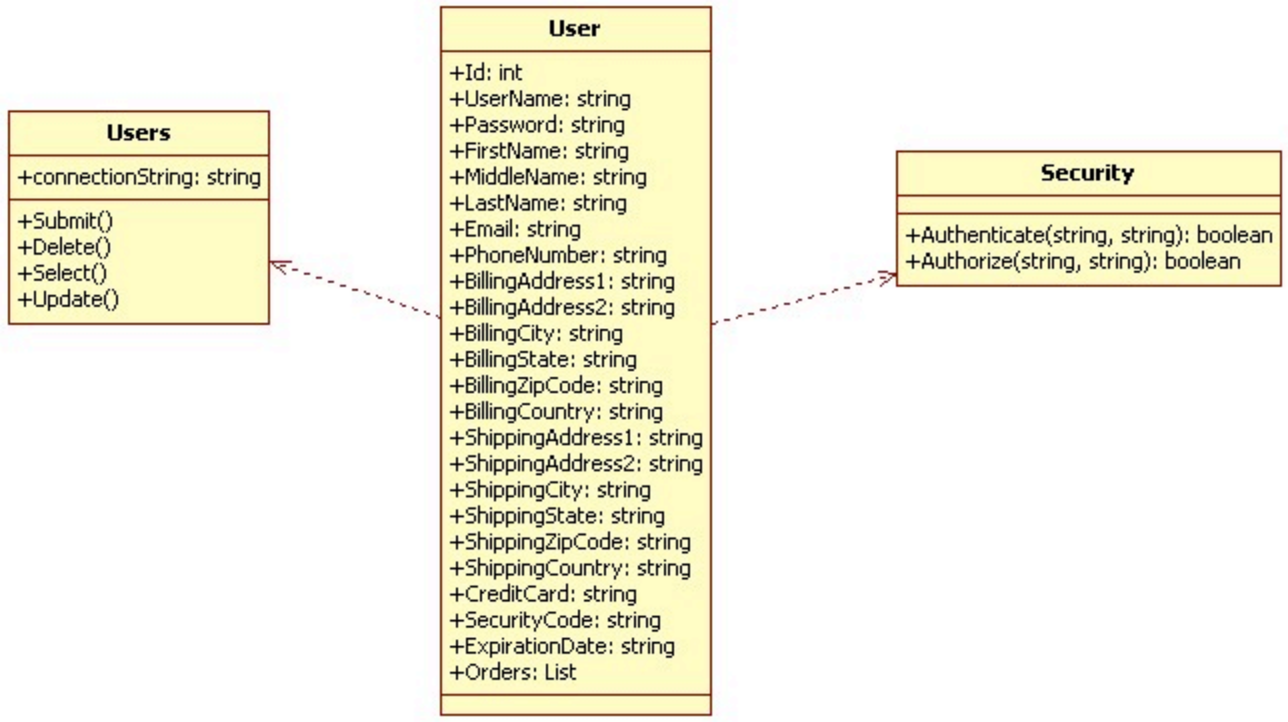


Attributes

- Id : int
  This is a public unique User ID number, used to manipulate particular User objects.
- UserName : string
  This is a public string, which holds the user name (name used to log-in) for each user.
- Password : string
  This is a public string, which holds the password (used to log-in) for each user.
- FirstName : string
  This is a public string that holds the first name for each user.
- MiddleName : string
  This is a public string that holds the middle name for each user.
- LastName : string
  This is a public string that holds the last name for each user.
- Email : string
  This is a public string that holds the email address for each user.
- PhoneNumber : string
  This is a public string that holds the phone number for each user.
- BillingAddress1 : string
  This is a public string that holds the first line of the billing address for each user.
- BillingAddress2 : string
  This is a public string that holds the second line of the billing address for each user.
- BillingCity : string
  This is a public string that holds the city of the billing address for each user.

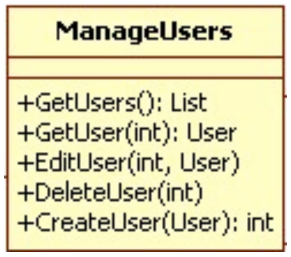
- BillingState : string
  This is a public string that holds the city of the billing address for each user.
- BillingZipcode: string
  This is a public string that holds the zip code of the billing address for each user.
- BillingCountry : string
  This is a public string that holds the country of the billing address for each user.
- ShippingAddress1 : string
  This is a public string that holds the first line of the shipping address for each user.
- ShippingAddress2 : string
  This is a public string that holds the second line of the shipping address for each user.
- ShippingCity : string
  This is a public string that holds the city of the shipping address for each user.
- ShippingState : string
  This is a public string that holds the city of the shipping address for each user.
- ShippingZipcode: string
  This is a public string that holds the zip code of the shipping address for each user.
- ShippingCountry : string
  This is a public string that holds the country of the shipping address for each user.
- CreditCard : string
  This is a public string that holds the credit card number for each user.
- SecurityCode : string
  This is a public string that holds the security code of the credit card for each user.
- ExpirationDate : string
  This is a public string that holds the expiration date of the credit card for each user.
- Orders : List
  This is a public List that holds the unique ID numbers of each order that the customer has placed.

Dependencies



Manage Users Class

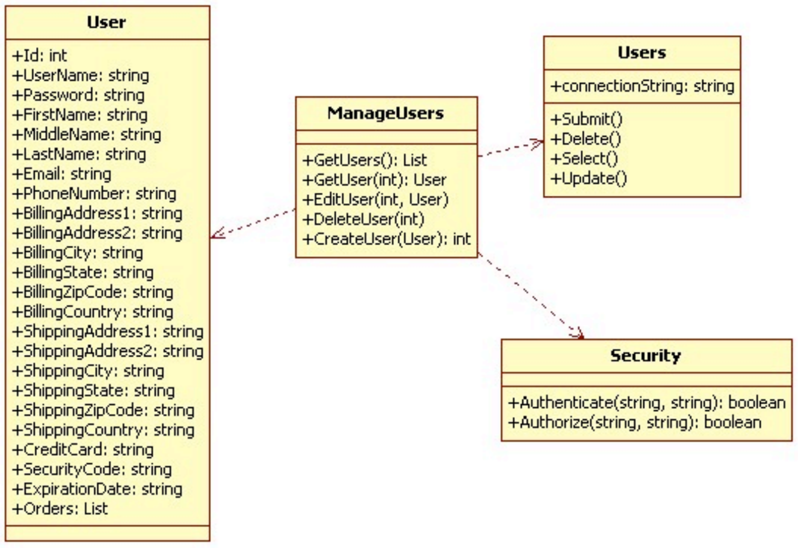
The ManageUsers class abstracts access to the User and Users classes. This allows user information and instances to be added, deleted, updated, or viewed. The ManageUsers class is as follows:



Operations

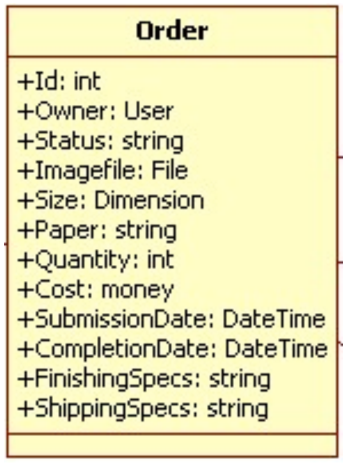
- GetUsers() : List
  GetOrders() is a public function which returns a list of all users. This is used for views that display all users, such as administration.
- GetUser(int) : User
  This is a public function which will return a specific User object, given a User.ID field. This is used for viewing a specific user's information.
- EditUser(int, User)
  This is a public function used to update a specific user's attributes, given a User.ID field.
- DeleteUser(int)
  This public function is used to permanently remove a user from the system.
- CreateUser(User) : int
  This is a public function that will be invoked to create a new user, used during the new-user registration process.

Dependencies



Order Class

The Order class represents a single order, holding all information relevant to a particular order which was placed by a customer.

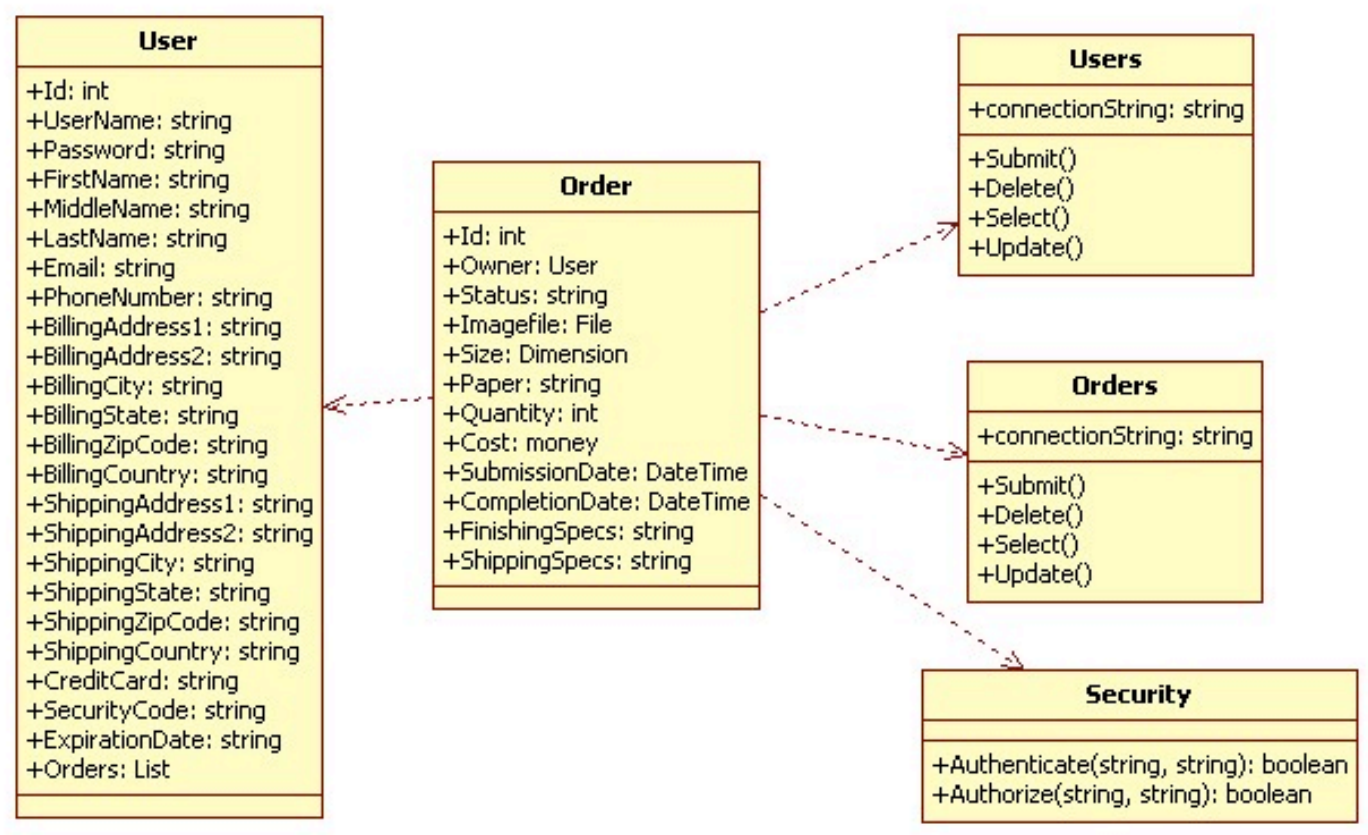


Attributes

- Id : int
  ID is an order identification number of type positive integer. It is a public attribute. It is used to identify the Order.
- Owner : User
  Owner is of type string. It is a public attribute. Owner is the name that identifies the User who created the order.
- Status : string
  Status is of type string. It is a public attribute. It displays the current status of the particular order.
- Imagefile : File
  Imagefile is of type file. It is a public attribute. It is the file uploaded by the customer while placing the order.
- Size : Dimension
  Size is of type Dimension. It is a public attribute. Size is the dimension Customer specifies for the print while placing the order.
- Paper : string
  Paper is of type string. It is a public attribute. Customer specifies the type of paper on which he wants the order to be printed.
- Quantity : int
  Quantity is of type positive integer. It is a public attribute. Customer specifies the number of copies of the print required at the time of the Order.
- Cost : money
  Cost is of type Money. It is a public attribute. Cost of the printing is displayed when the Order is placed.
- SubmissionDate : DateTime
  Submission date is of type DateTime. It is a public attribute. It records the time and date when the Order was placed.
- CompletionDate : DateTime
  Completion date is of type DateTime. It is a public attribute. The system specifies the time and date of when the Order would be completed based on the Order date.

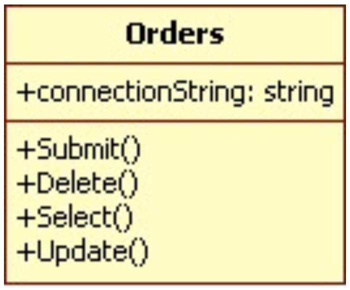
- FinishingSpecs : string
  Finishingspecs is of type string. It is a public attribute. The customer specifies the type of finishing required for the Order.
- ShippingSpecs : string
  Shippingspecs is of type string. It is a public attribute. The customer specifies the type of shipping required for the Order.

Dependencies



<u>Orders Class</u>

The Orders class represents the collection of Order objects that have been entered into the system. It allows access to the aggregate of the Order objects, useful for pages that display all orders and allow the user to select one.



Operations

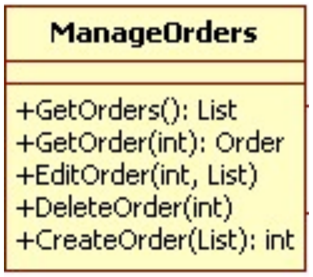
- Submit()
  Submit() is a public function which does not return any value. It is used to commit changes to Order in the database.
- Delete()
  Delete () is a public function which does not return any value. It is used to delete an Order in the database.
- Select()
  Select () is a public function which does not return any value. It is used to select an Order in the database.
- Update()
  Update () is a public function which does not return any value. It is used to Update changes to an Order in the database.

Dependencies

No Dependencies.

<u>ManageOrders Class</u>

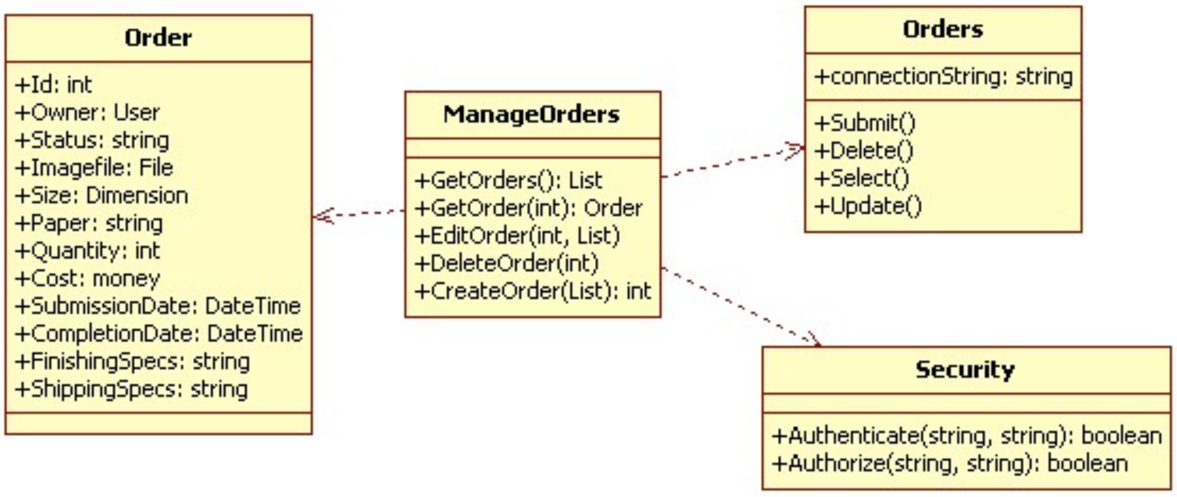
The ManageOrders Class allows an Employee to retrieve all Orders. The Administrator or the Customer Service can create an Order. However , the Administrator can edit or delete an order. The Manage Order Class is as follows :



Operations

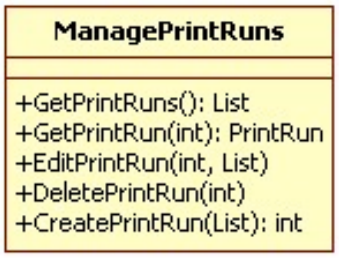
- GetOrders()
  GetOrders() is a public function which does not return any value. When an employee selects to retrieve the order list , the GetOrders() function returns the list of all the current Orders to be processed.
- GetOrder() : int
  Get Order is a public function which returns a value of type int. When an employee selects to retrieve a particular order, the GetOrder() function returns the particular order.
- EditOrder () : int
  EditOrder() is a public function which does not return any value. When this function is called it returns the list of Orders and a particular order can be selected to Edit accordingly.
- DeleteOrder(int)
  DeleteOrder() is a public function which returns a value of type int. When this function is called, the employee selects an order to delete and that particular order will be deleted.

Dependencies



<u>ManagePrintRuns class</u>

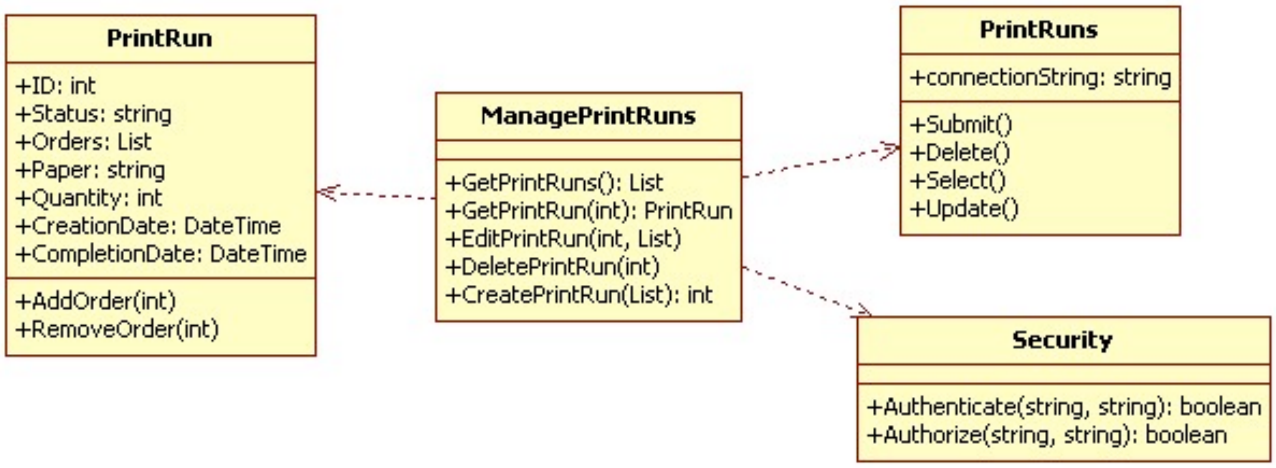
The ManagePrintRun class allows the administrator/worker manage PrintRuns, The worker can create and retrieve Print runs, and the administrator can edit and delete print run in additions of the capabilities of the workers.



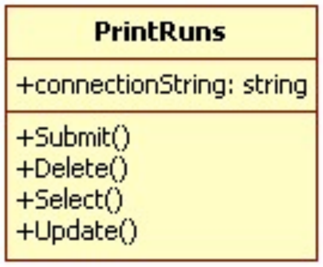
Operations

- GetPrintRuns() : List
  This funciton retrieves all the PrintRun in progress, return a list of current PrintRuns, then the worker can select one of them to see its contents or just verify it.
- GetPrintRun(int) : PrintRun
  This function retrieve a specific PrintRun, the parameter that the function require is the id of the printRun that the worker want to retrieve.
- EditPrintRun(int, List)
  This function modifies the attributes of a specific Run, the attributes are given to the function into a list of attributes then are set up and store into the Data Base.
- DeletePrintRun(int)
  This function delete a specific PrintRun, the parameter that it require is the PrintRun ID number.
- CreatePrintRun(List) : int
  This function create a new Print Run, return the ID number of the Print Run created.

Dependencies



<u>PrintRuns Class</u>



Operations

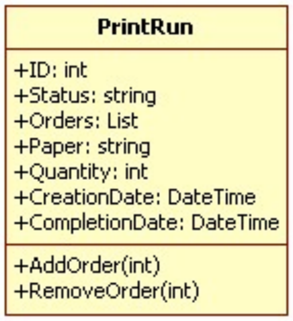
- Submit()
  Submit() is a public function which does not return any value. It is used to commit changes to PrintRun in the database.
- Delete()
  Delete () is a public function which does not return any value. It is used to delete a PrintRun in the database.
- Select()
  Select () is a public function which does not return any value. It is used to select a PrintRun in the database.
- Update()
  Update () is a public function which does not return any value. It is used to Update changes to a PrintRun in the database.

Dependencies

No Dependencies.

PrintRun Class

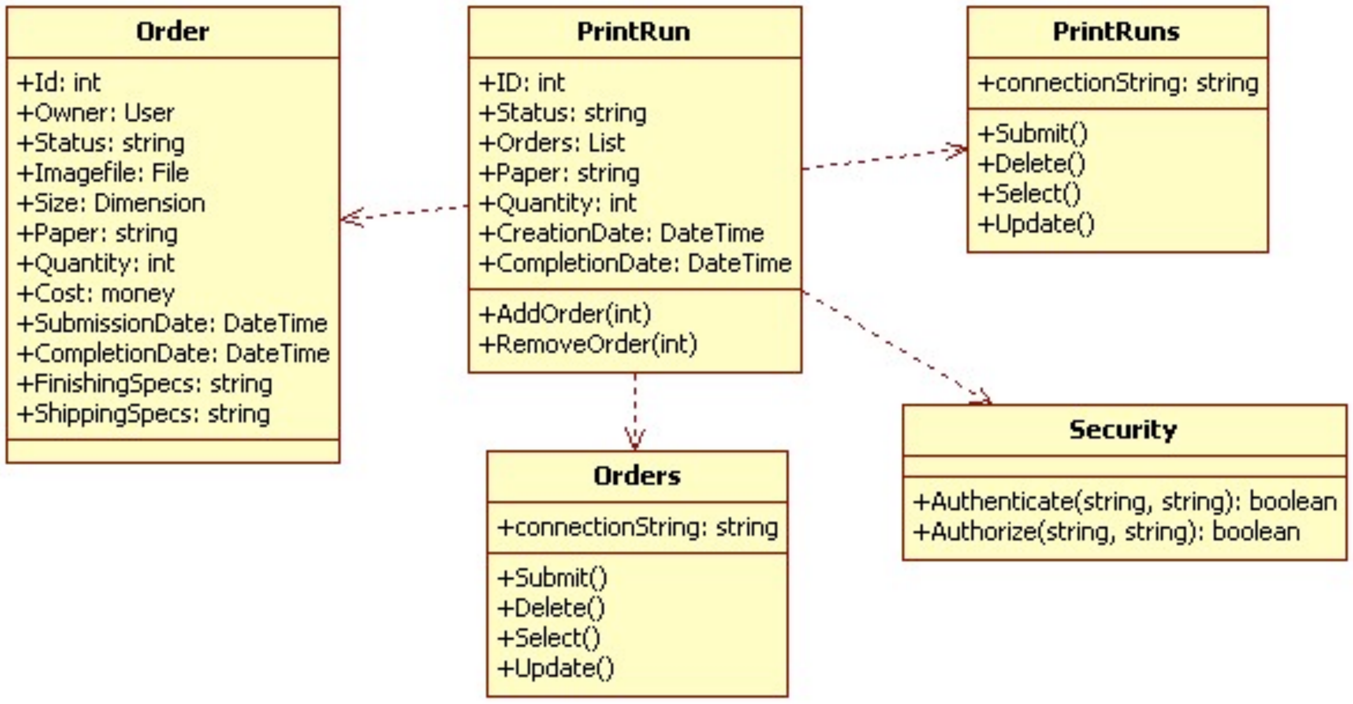
This Class is in charge about the operation directly related with PrintRun like add/delete Order to print run.



Attributes

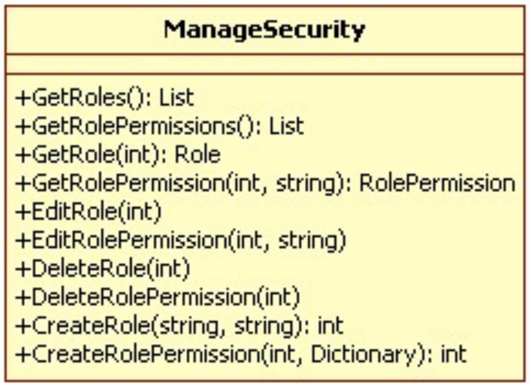
- Id : int
  The Id number of the actual PrintRun
- Status : string
  This refers to the current status (ready to print, printed, finished, shipped)
- Orders : List
  This is the list of orders that the PrintRun contents. to add a new order you need to use the function 'addOrder()'
- Paper : string
  This refer to the type and quality of the paper.
- Quantity : int
  This refers to the number of copy that need to be printed out.
- CreationDate : DateTime
  This is the Date when the Print Run is created
- CompletionDate : DateTime
  This refer to the Date in which the order will be ready to ship out.
- AddOrder(int)
  This function add a specific order into the PrintRun, the amount of orders that fit into a PrintRun depend on the size of each order. The worker has to take care about that.
- RemoveOrder(int)
  This function remove a specific order from the PrintRun. it requires the Order Id number.

Dependencies

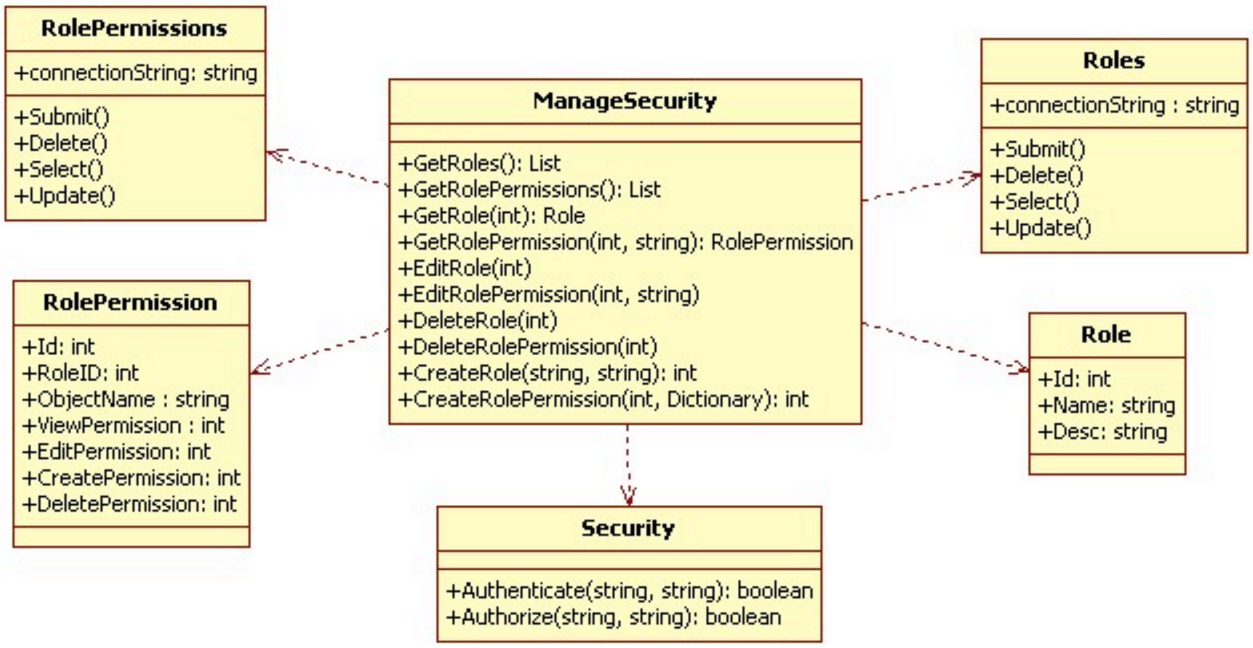


ManageSecurity Class

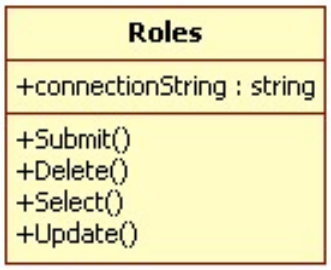
This class is responsible for encapsulating all the logic pertaining to Roles and RolePermissions.


ManageSecurity
+GetRoles(): List
+GetRolePermissions(): List
+GetRole(int): Role
+GetRolePermission(int, string): RolePermission
+EditRole(int)
+EditRolePermission(int, string)
+DeleteRole(int)
+DeleteRolePermission(int)
+CreateRole(string, string): int
+CreateRolePermission(int, Dictionary): int

Dependencies



Roles Class



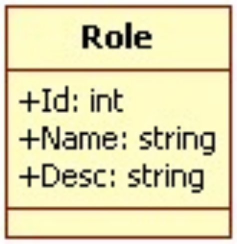
Operations

- Submit()
  Submit() is a public function which does not return any value. It is used to commit changes to Role in the database.
- Delete()
  Delete () is a public function which does not return any value. It is used to delete an Role in the database.
- Select()
  Select () is a public function which does not return any value. It is used to select an Role in the database.
- Update()
  Update () is a public function which does not return any value. It is used to update changes to an Role in the database.

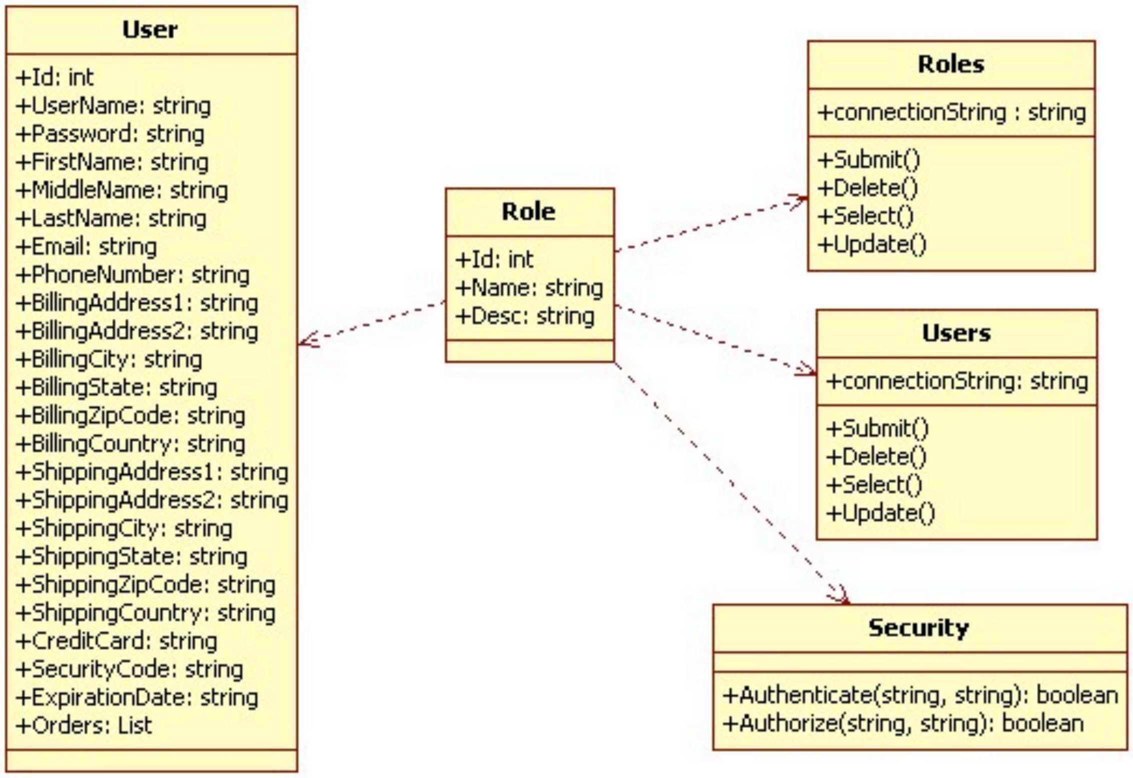
Dependencies

No Dependencies.

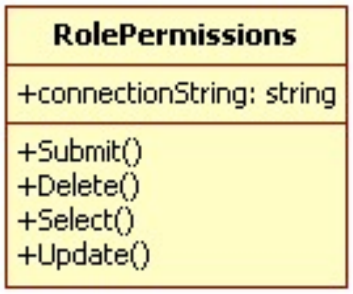
Role Class



Dependencies



<u>RolePermissions Class</u>



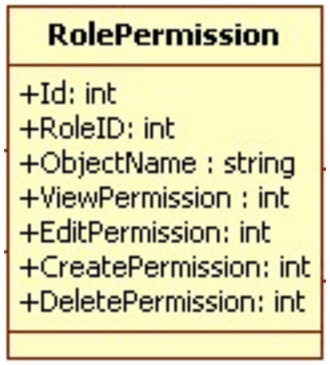
Operations

- Submit()
  Submit() is a public function which does not return any value. It is used to commit changes to RolePermissions in the database.
- Delete()
  Delete () is a public function which does not return any value. It is used to delete a RolePermissions in the database.
- Select()
  Select () is a public function which does not return any value. It is used to select a RolePermissions in the database.
- Update()
  Update () is a public function which does not return any value. It is used to Update changes to a RolePermissions in the database.

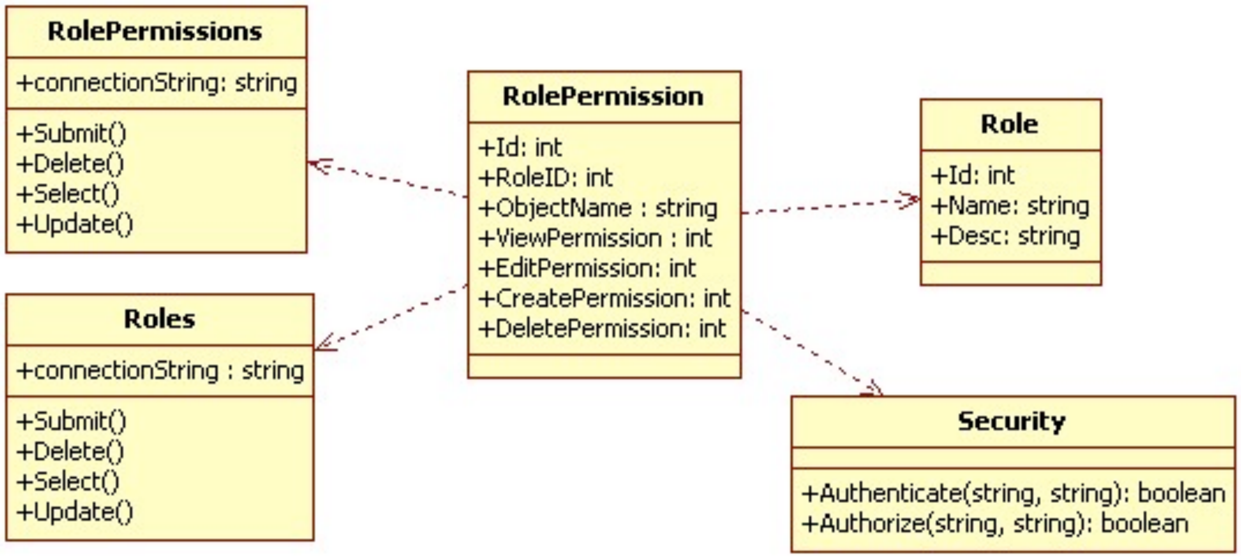
Dependencies

No Dependencies.

<u>RolePermission Class</u>

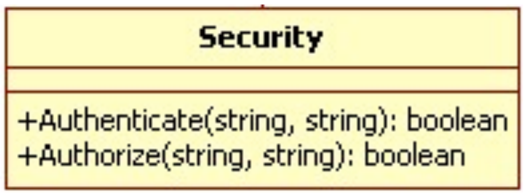


Dependencies



Security Class

This class will be used when a user attempts to Login in order to authenticate the user. This class will also provide the mechanism for all UI objects and Manager objects to be sure a user is authorized to perform a particular action on a particular object.


Security
+Authenticate(string, string): boolean
+Authorize(string, string): boolean

Dependencies

No Dependencies.

4. Glossary

- **Administrator** : A member of the company, who has all the rights of a regular Employee plus other administrative rights such as deleting a user, editing a user's information, etc.
- **Customer**: A client of the company, who can submit orders for printing, pay those orders, and track the orders as well.
- **Company**: Specifically, XYZ Printing Co.
- **Customer Service** : A member of the company who can take an order on behalf of a customer – to act as a proxy for an offline customer.
- **Worker**: A member of the company, who has all the rights of any User plus other rights such as process customer orders, create print runs, etc.
- **Finishing**: The part of the company workflow where the cutting and resizing process is taking place.
- **Order**: A User can create an order and save it into the system, which contains specifications regarding printing details, a file to be printed, and payment information.
- **Portal**: Web-based interface presented to customer and employees.
- **Printing**: The part of the company workflow where the print-manufacturing process is taking place.
- **Print Run** : A single file created by an employee, which is sent to printing.
- **System**: PWAS is considered the system, and it entails all the software that takes care of the workflow management.