Printshop Workflow Automation System
P.W.A.S

Deliverable #2
Design Document

Presented by:

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

1. Introduction

1.1 Overview of the system

XYZ Printing Co. (Henceforth referred to as "the customer") is a printing company specializing in commercial gang runs (See 1.4 for a complete explanation on this). The customer is currently experiencing a surge in production, but finds itself frequently stumbling due to human caused errors. It was determined that most errors stem from their complicated order acceptance system, as well as decentralized and unorganized production workflow. The proposed system is meant to address these problems, allowing the customer to do more with the current resources that they have.

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 modified 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 available at this time.

1.2 Purpose of the system

To facilitate and automate production for the customer. To provide a uniform interface for customer order submission, order tracking, employee workflow, and management activities that will enhance productivity and efficiency.

1.3 Design goals

Throughout development, it must be kept in mind by all teams that the main goal of this system is to make production more efficient for the customer. All design and development decisions must be centered around this goal. The system must be robust, reliable and intuitive for its users, so that all processes can be accomplished quickly. The interface must be clean, easy to understand, and convey all necessary information to the users. The project will be considered a success if the customer is able to speed up production to one business day on all orders.

1.4 Definitions, acronyms, and abbreviations

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

1.5 References

It is recommended that all teams involved in the project read the following documents so that more familiarity with the problem domain is achieved:

- http://www.howstuffworks.com/offset-printing.htm
- http://en.wikipedia.org/wiki/Gang_run_printing
- .NET Framework Design Guidelines: http://msdn.microsoft.com/en-us/library/ms229042.aspx

2. Current software architecture

Currently, there is no dedicated software in use by the customer. The current system consists of a number of procedures and non-connected systems that have been created ad-hoc to cope with a steady increase in production. Orders are accepted through e-mail, ftp, actual delivery of media, etc. Work is sorted and scheduled manually, and orders are assigned verbally. When job owners need to track an order, they must call the plant directly, and somebody must physically go to the production area to check the status of the order.

Direct comparison to other existing systems is not possible at this time, since similar systems are not available commercially (There are a few commercial systems, but none that could be found offers a comprehensive solution; instead they only focus on particular problems areas.) It is known that complete systems exist, but these are proprietary, and closely guarded trade secrets. Our proposed system architecture arises from a thorough analysis of the customer's needs, and desired goals, rather than imitation of other programs and systems.

3. Proposed software architecture

The proposed system will implement a three-tiered architecture, separating interface, program logic and storage. The main reason for our 3-tier architecture selection is because we want a thin client that doesn't require installation. It makes the system location-independent so we can have the customer anywhere in the world where there is an internet connection, without the need to carry around an installed version of the client.

3.1 Overview

The interface to the system will be presented through a web browser. There will be no other way for users to interact with the system. The logic layer consist of the web server, which will be hosted in-house by the customer, it will process all calls from the interface, as well as storing and retrieving data on the storage layer. This last layer will consist of a Database Engine, and a wrapper class.

3.2 Subsystem decomposition

The PWAS system utilizes a modified three-tier design, comprised of: the interface, application logic, and storage (Data Access and Data Storage). In this case, the interface is the PWASWebClient, a standard web-browser, communicating with the PWASWebServer via http. The PWASWebServer is a standard web-server, driven by custom application logic. This PWASWebServer will interface to a storage system (realized by a database) to store persistent objects via ODBC. The interface to the storage system is flexible – it can be pointed at a variety of data sources, such as files or web-services without requiring a change in any of the top tiers.

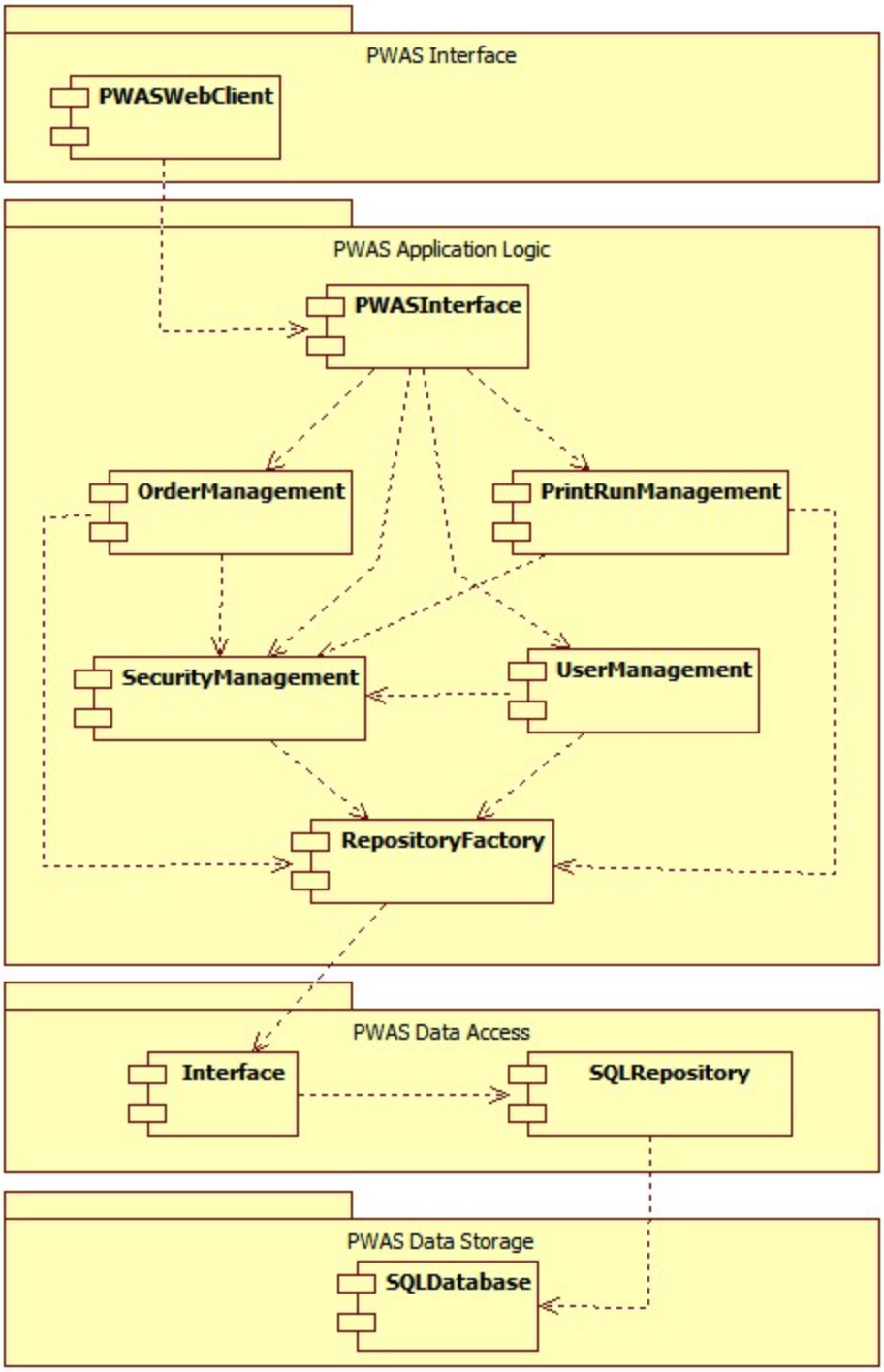


Figure 1 – Subsystem Decomposition for PWAS

PWAS Interface Tier:

PWASWebClient - The PWASWebClient is a Web browser, running on the end-user's machine. This provides the graphical interface between the user and the system. This subsystem communicates with the PWASWebServer subsystem, to receive html files, which it renders into a user-friendly graphical interface.

PWAS Application Logic Tier:

PWASInterface - This subsystem is responsible for generating the web-page for each user request and contains the code to handle the multiple asynchronous network connections for each simultaneous customer connection. This subsystem communicates with the PWASWebClient and various entity management subsystems to generate a html file and send it to the appropriate user over the network (an instance of the PWASWebClient subsystem).

OrderManagement - This subsystem deals with all aspects of customer orders, such as viewing and editing order information. This subsystem is primarily used by the PWASInterface subsystem, and communicates with the SecurityManagement subsystem to authorize access to customer order objects.

PrintRunManagement - This subsystem deals with all aspects of PrintRun objects, such as viewing and editing PrintRun information. This subsystem is primarily used by the PWASInterface subsystem, and communicates with the SecurityManagement subsystem to authorize access to PrintRun objects.

UserManagement - This subsystem deals with all aspects of User objects, such as viewing and editing User information. This subsystem is primarily used by the PWASInterface subsystem, and communicates with the SecurityManagement subsystem to authorize access to User objects.

SecurityManagement - The SecurityManagement subsystem has two primary functions: to authenticate users (by checking their username and password combination) and to authorize access to other subsystems. For example, the OrderManagement subsystem will call the SecurityManagement subsystem whenever a request is made to edit a order. If that user's role allows them this access, the OrderManagement subsystem will make the update, otherwise, the update is denied.

RepositoryFactory – The RepositoryFactory abstracts access to the PWAS Data Access tier, and interacts with the Interface subsystem in that tier. It is responsible for providing a uniform interface to the data from the PWAS Data Access tier, regardless of data source.

PWAS Data Access:

Interface – The Interface subsystem's primary responsibility is to provide a uniform and consistent communication source for the RepositoryFactory. The Interface may use a variety of different data sources to retrieve data from the PWASDataStorage tier – in this case, we are

using a SQL Server instance. If this database needs to be changed in the future, the PWAS Data Access tier is the only tier that is affected.

PWAS Data Storage:

SQLDatabase - The SQLDatabase subsystem is indirectly used by the OrderManagement, PrintRunManagement, UserManagement, and SecurityManagement subsystems to manage persistent data objects, via the PWAS Data Access tier. This subsystem can be used to retrieve, update, or add new persistent data objects, such as Users, Orders, or PrintRuns. The SecurityManagement subsystem uses it to look up username and password combinations, and retrieve information about which user roles can access which functionality

3.3 Hardware/software mapping

There are three types of nodes that will host various components of the system: the ClientMachine, the WebServer and the Database. These nodes are one-to-one mappings of each tier in the traditional three-tier design, described in section 3.2.

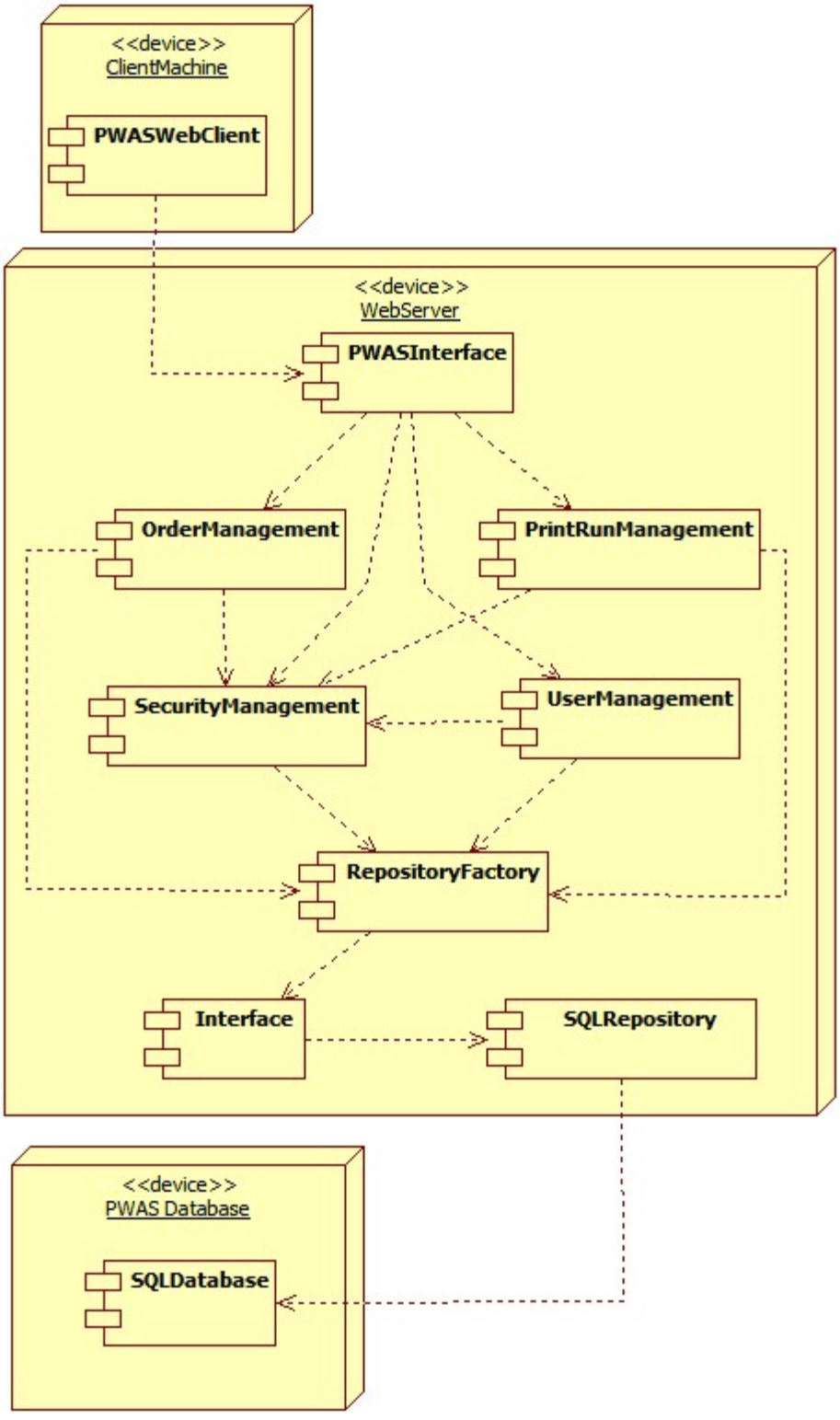


Figure 2 – Hardware to Software Mapping for PWAS

The ClientMachine hosts the PWASWebClient, which is a standard supported web-browser, such as Internet Explorer 8 or Firefox 3.5. Typically, this is a end-user's internet-connected personal computer or a company-owned PC that is being used by a worker. This is the physical embodiment of the PWAS interface tier.

The WebServer hosts several subsystems: PWASInterface, OrderManagement, PrintRunManagement, UserManagement and SecurityManagement. Working in concert, as directed by the business logic in the PWASInterface, these subsystems communicate with the persistent storage solution via the RepositoryFactory, Interface, and SQLRepository. Together, these subsystems comprise the application logic tier. This machine is owned by the company and exists on a internal network. It communicates with the ClientMachine via http, and the Storage machine with ODBC.

The Database hosts the SQLDatabase subsystem. This node is independent of the WebServer, but may co-exist on the same machine. Its design-independence allows it to to be hosted elsewhere such as a separate virtual / physical machine, should the need arise.

3.4 Persistent data management

Storage Strategy:

The system will maintain its data using a Relational Database, which will be managed with a Database Management System (DBMS). The DBMS will take care of concurrency and synchronization issues regarding the accessibility of the persistent data.

Persistent Objects:

- Order : Will maintain all the information regarding an Order object
- User : Will maintain all the user information that's part of the user profile such as name, address, and billing information, shipping information, etc.
- Run : Stores all the information regarding a Run
- OrdersToRuns Association : Will maintain a relation between Orders and Runs, to associate each Run with the Orders that are part of that Run.
- Roles : Maintains a definition of Roles, as well as an id, and a description for those Roles.
- RolePermissions : Maintains an association between the roles, the objects that they can act upon, and the actions that can be applied to those objects.

3.5 Access control and Security

Upon registration, each user is assigned a role type by the System Administrator. These roles are defined by the Administrator along with the privileges that members of this role type have for system objects. The roles and their respective permissions are stored in the database.

There are four basic role types that are defined by default. These are: Customer, Customer Service, Worker, and Administrator. The objects for which permissions are defined include: Users, Orders, Print Runs, and Roles. The access matrix below summarizes the default actions that each of the role types are authorized to perform on each of the system objects.

Object Actor	User	Order	Run	Roles
Customer	<<create>> Edit profile	<<create>> Pay Track		
Customer Service	<<create>> Edit profile	<<create>> Pay Track		
Worker	<<create>> Edit profile	Add to run Update status View	<<create>> Edit Update status	
Administrator	<<create>> Edit profile Manage users	Manage orders		<<create>> Edit permissions

Figure 3 – Access Matrix for PWAS

As part of the registration process, users will be required to provide a user name and password which will serve for future authentication to the system. All data transmissions will be carried over a secure HTTPS connection to ensure the privacy of personal customer information, such as profile information and order information.

The controls and buttons that a user sees on the interface are subject to the user's access authorization. A user is not presented with controls or options that are beyond their access scope. In addition to this, all actions are authorized again at time of execution to ensure that users are not trying to access views or controls outside of their access scope.

The following is a more detailed explanation of the roles and role permissions of PWAS:

Role Ids:
1 - customer
2 - customer service
3 - worker
4 - administrator

Permissions:
0 - no access
1 - own
2 - all

permissionID	roleID	object	obj_update	obj_view	obj_create	obj_delete
1	1	user	1	1	1	0
2	1	order	1	1	1	1
7	1	run	0	0	0	0
8	1	role	0	0	0	0
9	2	user	2	2	1	0
10	2	order	2	2	2	2
11	2	run	0	0	0	0
12	2	role	0	0	0	0
13	3	user	1	1	1	0
14	3	order	2	2	0	0
15	3	run	2	2	2	2
16	3	role	0	0	0	0
17	4	user	2	2	2	2
18	4	order	2	2	2	2
19	4	run	0	0	0	0
20	4	role	2	2	2	2

Figure 4 – Roles and Role Permissions for PWAS

3.6 Global software control

The internal control flow of PWAS is event-driven. This is because the web server objects wait for requests from the web browser. When a request is received, the web server processes it and dispatches it to the appropriate page controller. We use page controllers to realize the boundary and control objects of PWAS. A preprocessor then generates views from the different page controllers. These controllers then invoke methods on entity objects and storage objects to allow for the functionality of our system.

3.7 Boundary conditions

The starting, shutdown and installing of the PWAS define the boundary conditions.

Installing: Since PWAS is a web-based application, it does not need explicit installation execution. Instead PWAS files need to be copied to the WebServer.

Starting: The Administrator starts up the WebServer service making the PWAS system available to customers/workers. At this point the customers can connect to PWAS system by opening a web browser with PWAS web page address.

Shutdown: The Administrator shuts down the WebServer's service.

Exception Handling: System maintenance will be done on weekends, between 12am and 7am, occurring less than twice per month and during this period the WebServer services will be shut down.

4. Subsystem services

UserManagement Subsystem: This subsystem is responsible for managing different users of the system by taking care of the login information of different users. It provides functions for Register, Log in and Edit accounts. It manages the usernames and passwords of all users of the system for security purposes. This subsystem uses the services of storage subsystem to store and retrieve login information. System administrator and all users of the system communicate with this subsystem.

The operations provided by this subsystem are:
- Register
- Login / Logout
- Edit Profile

OrderManagement Subsystem: This subsystem is responsible for managing orders. It provides functions for Creating, Editing and Saving an Order. The customer can create and save an order. The Administrator has the option of editing the Order information before it is submitted to Printing. This subsystem uses services of SecurityManagement subsystem to authorize access to editing orders for the User.

The operations provided by this subsystem are:
- Create Order
- Edit Order
- Save Order

PrintRunManagement Subsystem: This subsystem is responsible for managing PrintRun. It provides functions for creating and editing PrintRun. This subsystem uses the services of User Interface and SecurityManagement subsystem to authorize access to the User.

The operations provided by this subsystem are:
- Create Run
- Edit Run

SecurityManagement Subsystem: This subsystem has two primary functions: to authenticate users (by checking their username and password combination) and to authorize access to other subsystems.

The operations provided by this subsystem are:
- Authenticate
- Authorize

5. 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.

5.1 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

6. Packages

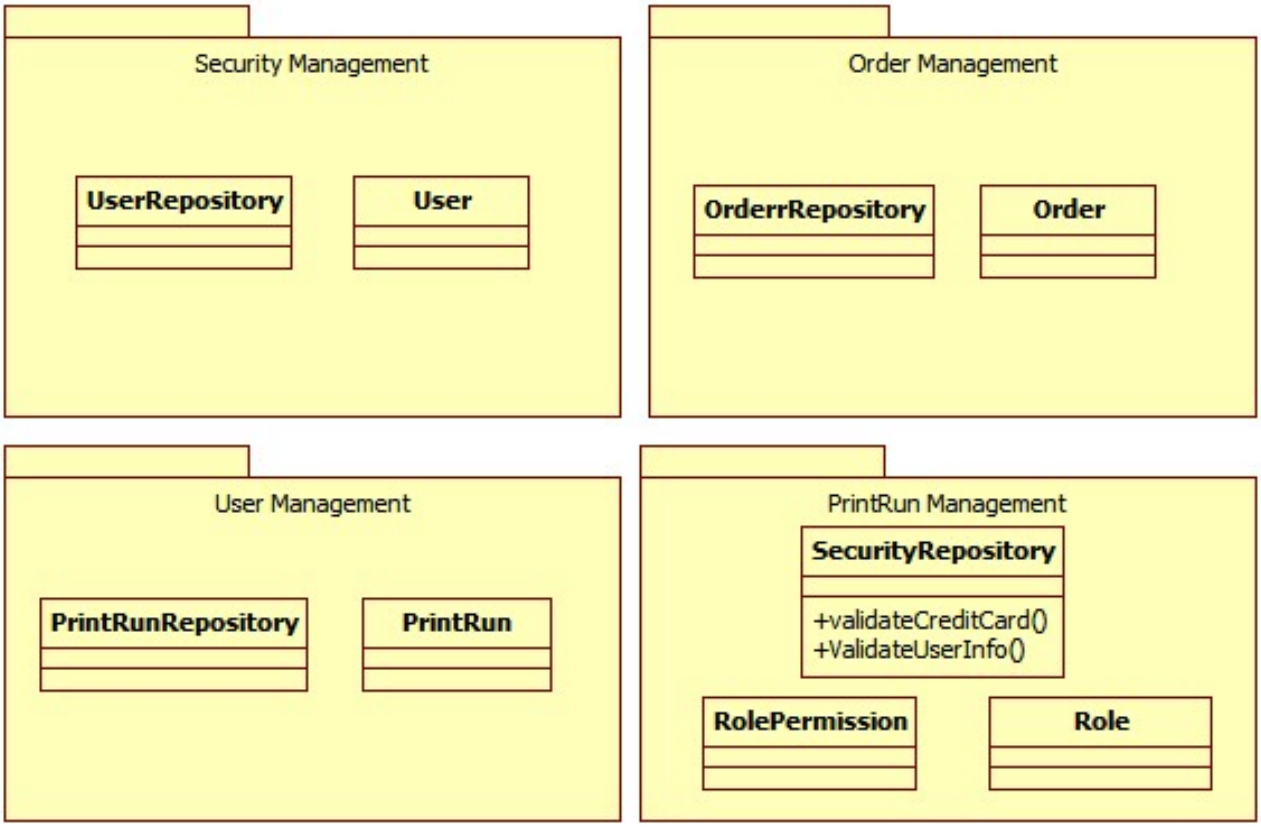


Figure 5 – Packages of PWAS

User Management Package

Filename	Classes contained
ManageUsers.cs	ManageUsers
User.cs	User

Overview

The User Management Package consists of ManageUsers 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. 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
Order.cs	Order

Overview

The Order Management Package consists of the ManageOrders and Order classes. This package encapsulates all the objects necessary to manage an order. 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.

PrintRun Management Package

Filename	Classes contained
ManagePrintRuns.cs	ManagePrintRuns
PrintRun.cs	PrintRun

Overview

The PrintRun Management Package consists of the ManagePrintRuns 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. 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
Role.cs	Role
RolePermission.cs	RolePermission

Overview

The Security Management Package consists of the ManageSecurity, Role, 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. 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.

7. Class interfaces

The detail documentation of all the classes, packages, methods, and attributes will be attached in HTML format. These HTML files are the output of a tool which generates documentation based on the actual code in place. Below we outline an overview of all the classes, as well as a more detailed view of the dependencies of each classes. The automatically generated documentaiton based on the code can be referenced if more information is needed.

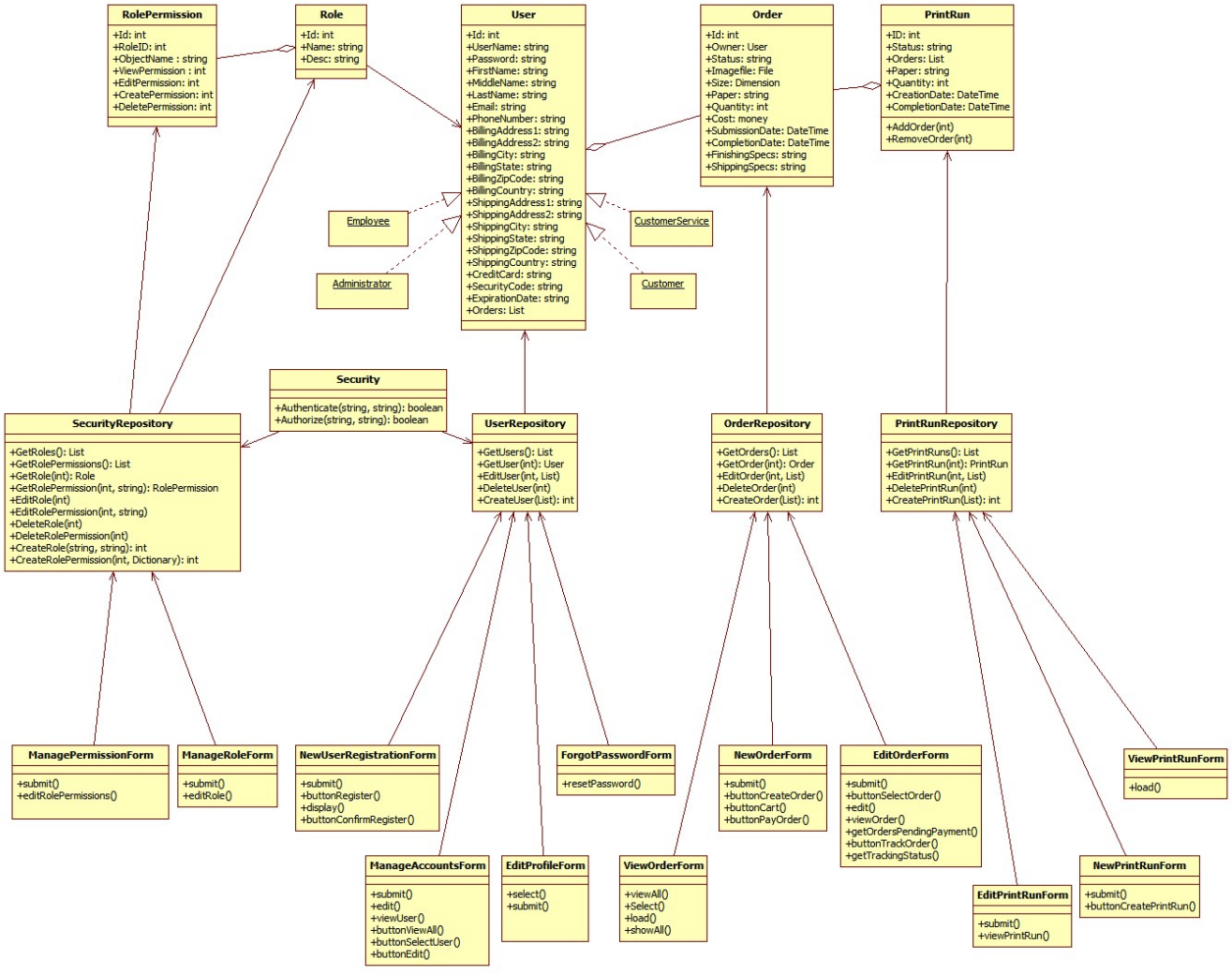
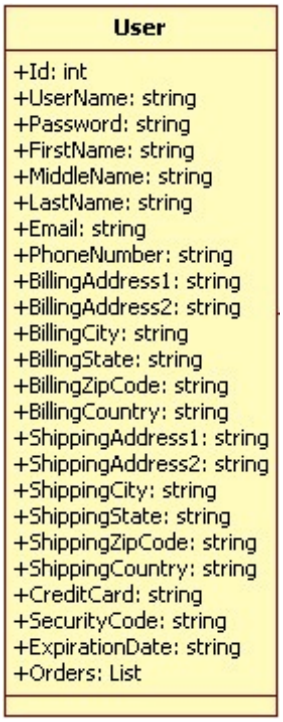


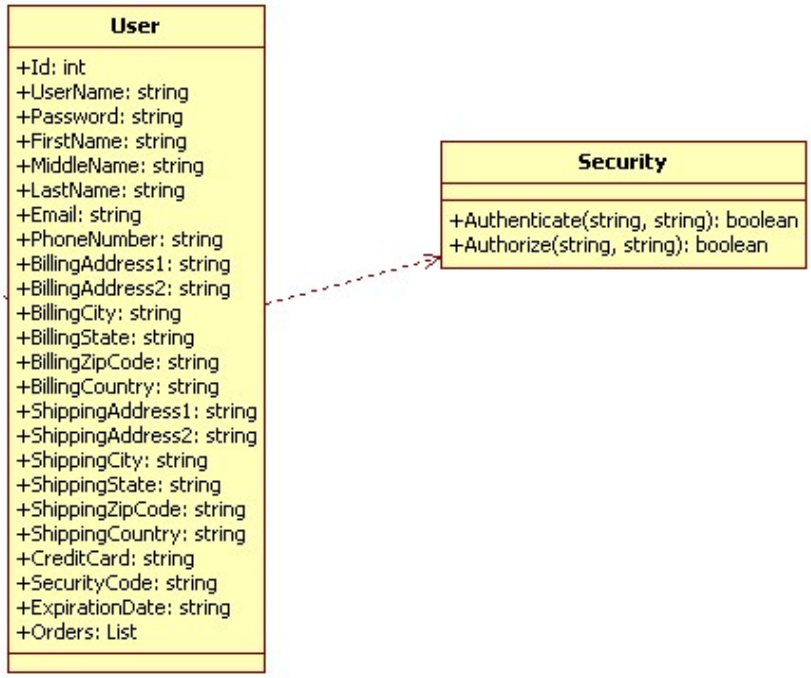
Figure 6 – Main Object Diagram for PWAS

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, worker, customer, or customer service.

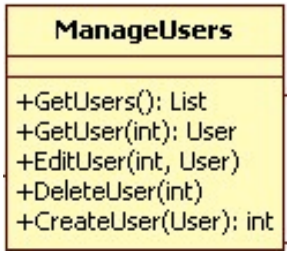


Dependencies

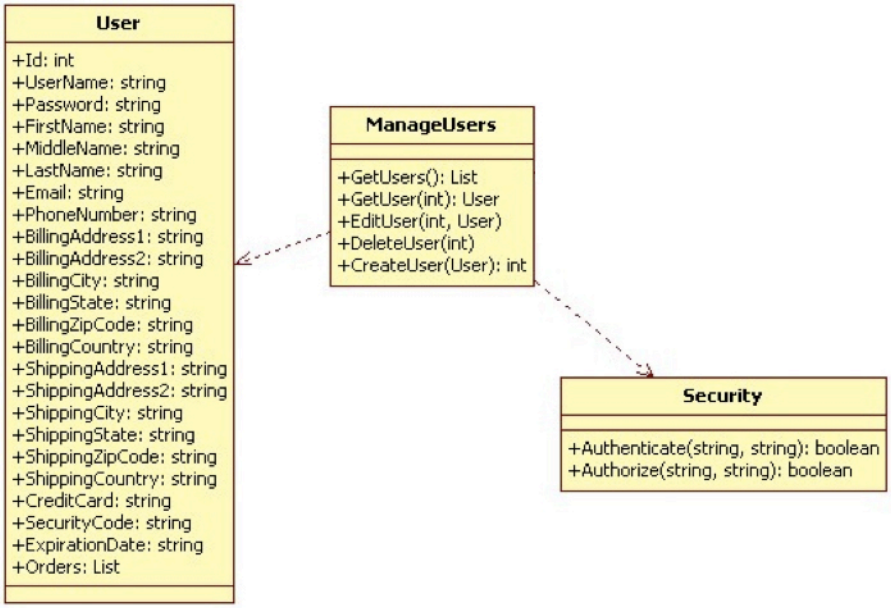


Manage Users Class

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:

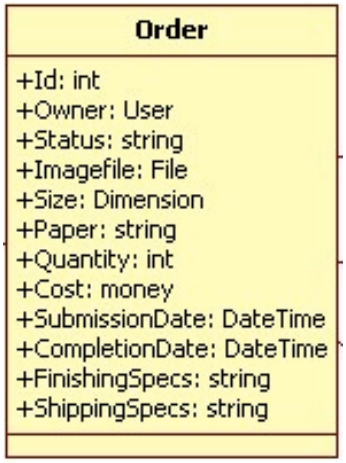


Dependencies

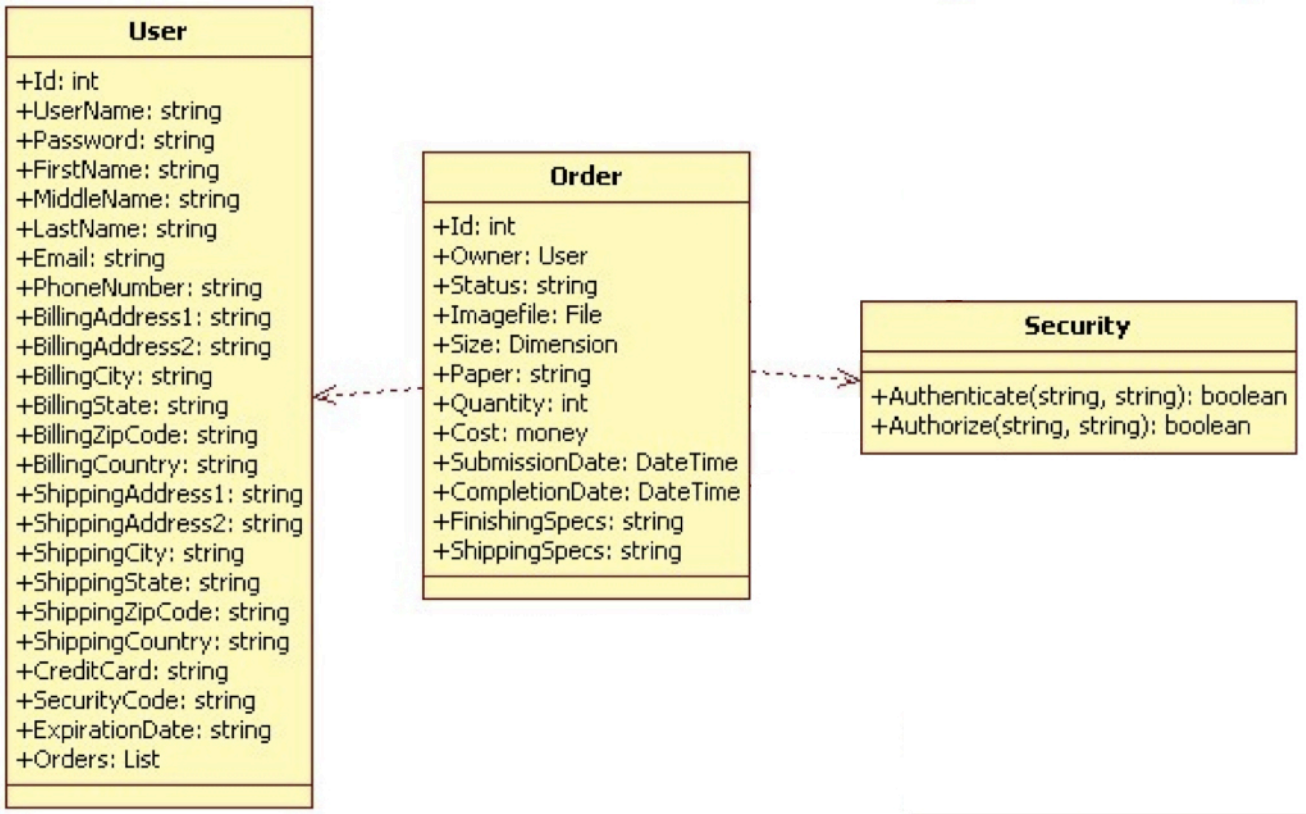


Order Class

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

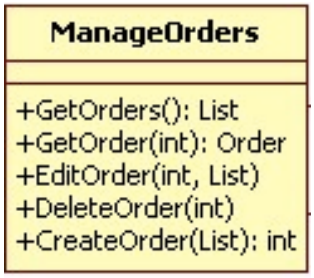


Dependencies

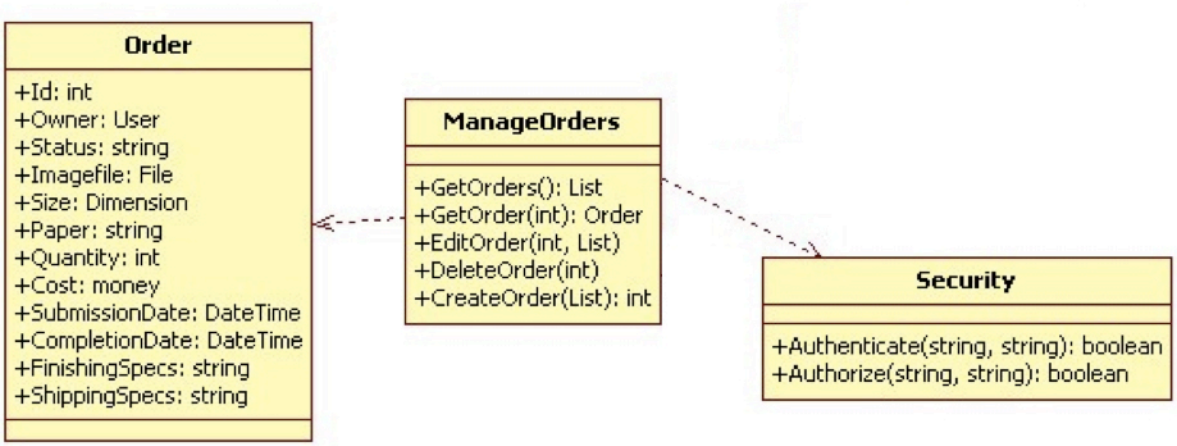


ManageOrders Class

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 :



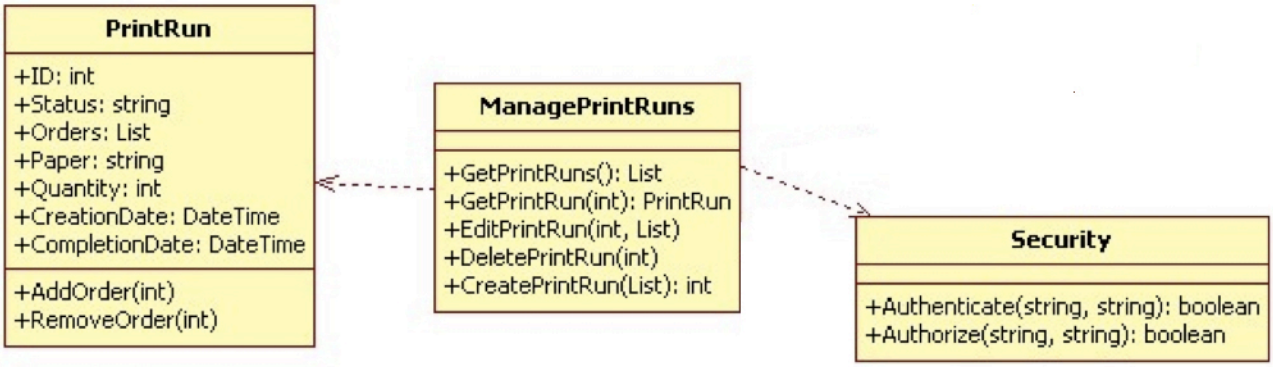
Dependencies



ManagePrintRuns class

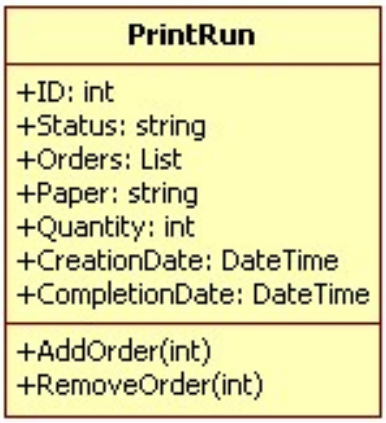
The ManagePrintRun class allows the administrator/worker manage PrintRuns, The worker can create and retrieve Print runs, and the administrator can edit and delete print run in additions of the capabilities of the workers.

Dependencies

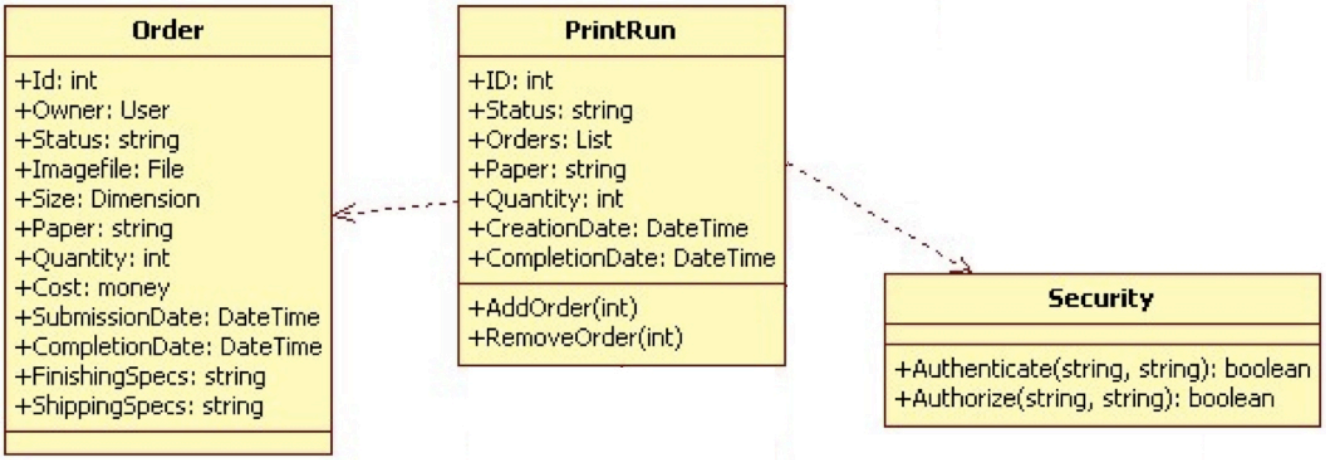


PrintRun Class

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

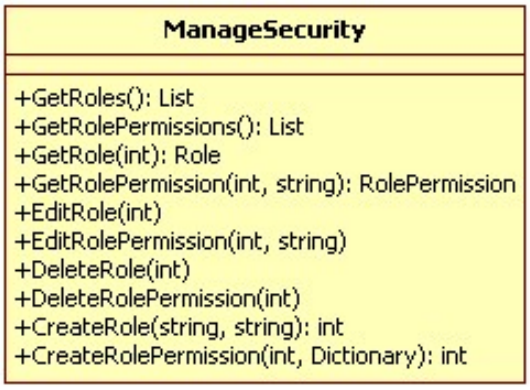


Dependencies

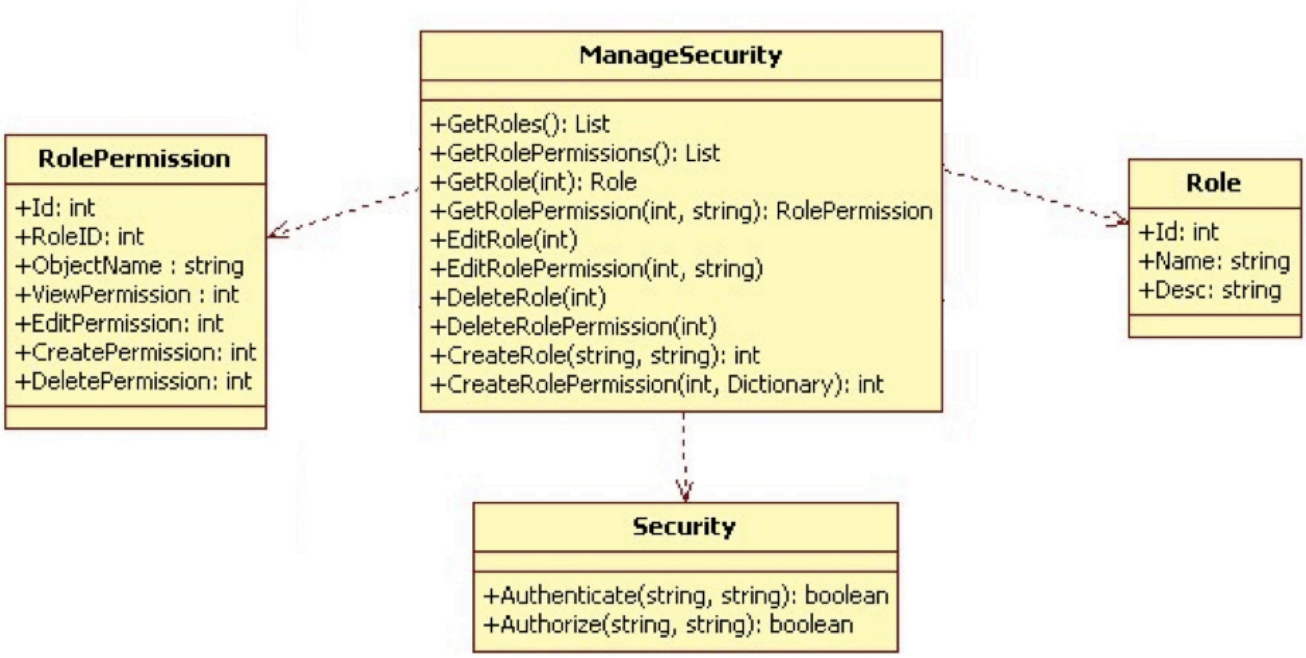


ManageSecurity Class

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



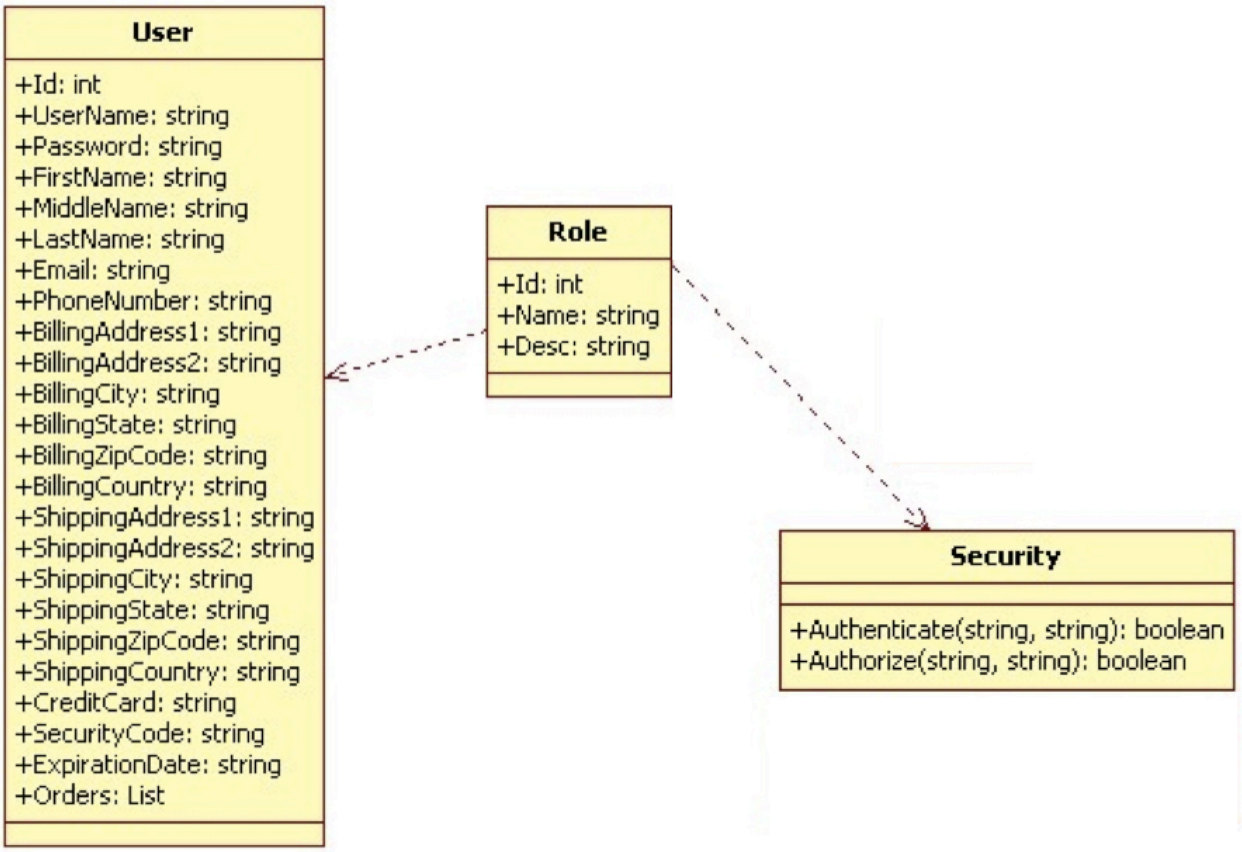
Dependencies



Role Class

The Role class is responsible for keeping the role definition which is used by Users.

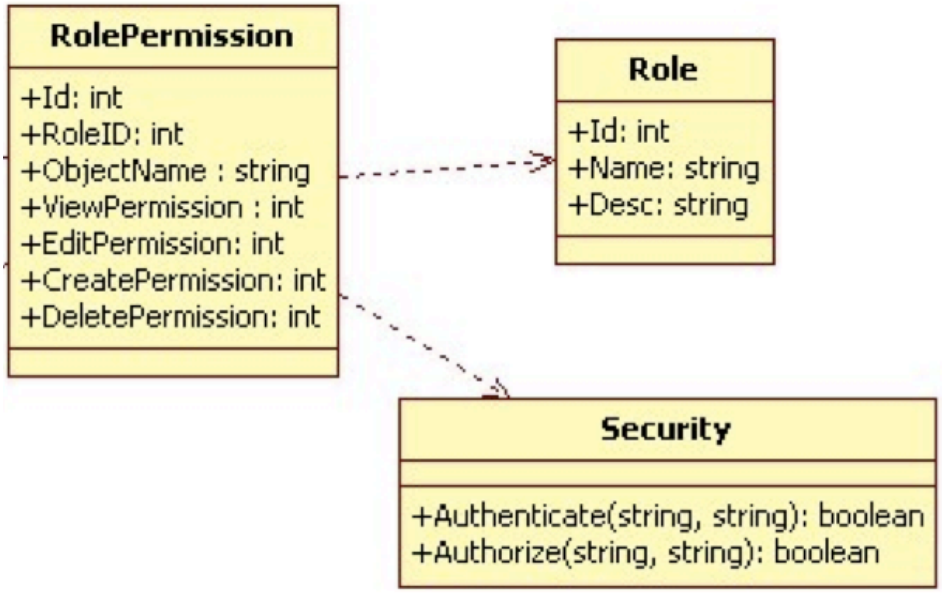
Dependencies



RolePermission Class

The Role Permission class maintains a definition of the permission that each role has on diferent objects.

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.

Dependencies

No Dependencies.

8. 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.