



A Lockheed Martin Company

IT SERVICE REQUEST PORTAL, *PROJECT REPORT*

COMPUTER SCIENCE DEPARTMENT, CSU LOS ANGELES

ANDREW AHN, ZAINAB ALZAYER, KEVIN HUYNH, RAYMOND WU, IGNACIO ZUNIGA

6/12/2016

DOCUMENT REVISION HISTORY

Release No.	Date	Revision Description	Created By	Approved By	Approved On
Rev 0	6/6/2016	First draft	Andrew Ahn, Zainab Alzayer, Kevin Huynh, Raymond Wu, Ignacio Zuniga	Dr. Chengyu Sun	
Rev 1	6/12/2016	Second draft	Andrew Ahn, Zainab Alzayer, Kevin Huynh, Raymond Wu, Ignacio Zuniga	Dr. Chengyu Sun	
Rev 2					
Rev 3					

DOCUMENT REVIEW AND APPROVAL BY IT

Function/Role	Name	Review Date	Approval Date
DEV			
QA			
INF			

TABLE OF CONTENTS

1 Introduction	5
2 System Architecture	7
3 Design and Implementation.....	10
3.1 SharePoint: Lists and Workflow	10
VPN Request List	11
Approvers List	15
Task List.....	15
Workflow	16
3.2 IT Service Portal User Interface	18
My Request Page	18
Review Request Pages	19
Submitting New Requests.....	19
User Interface Videos.....	20
3.3 Connecting ASP.NET and SharePoint	21
ASP.NET Model Classes.....	21
Controllers	22
3.4 Running the ASP.NET Application	23
4 Conclusions	24

1 INTRODUCTION

QTC, the customer, is the largest private provider of government-outsourced occupational health and disability examination services in the nation. QTC employees provide services across all 50 US states and 5 territories. QTC is currently utilizing several legacy forms and outdated associated workflows. Many of these forms and workflows result in significant inefficiencies within QTC Operations staff as well as in supporting IT staff.

QTC IT is currently utilizing several legacy forms and outdated associated workflows. Many of these forms and workflows result in significant inefficiencies within QTC Operations staff as well as supporting IT staff. The IT Service Request Portal is a system that centralizes the IT request process. The user-facing end is a web application. The web app contains functionality for request creation, tracking, and approval. The web app contains supplementary content, in the form of an FAQ, for each request type.

The project objective is to create a self-service portal for common QTC Support/System Access Requests along with electronic workflow to streamline each request type. The IT Catalog will replace paper forms with a web portal. The system will contain persistent records on all service requests. The system will contain a workflow that automatically initializes for each service request instance. The workflow will persist for all stages of each service request instance: submitted, pending, rejected, and accepted. The workflow will generate emails to alert requestors, approvers, and IT staff of changing request statuses. The IT Service Catalog will provide a space for requestors to check the status of each service request. The catalog will provide a space for requestors, approvers, and IT staff to see each request.

The system is designed with many request types in mind; the implementation can be extended to include other request types. At this time, the IT Service Request Portal contains functionality and content for one request type: VPN Remote Access Request. When extending the implementation, changes must be made to both the front end and the back end. All development is done in SharePoint and ASP.NET. A working implementation of the VPN Remote Access Request was reached within three months by the CSULA development team. According to QTC liaisons, the VPN Remote Access Request type is the most involved request type. The implementation of other request types should take successively less time.

Deployment of the IT Service Catalog requires two steps: SharePoint must be deployed separately. Then the ASP.NET portion can be deployed. The deployment of the SharePoint portion is not straightforward. Due to SharePoint 2010 limitations, the workflow cannot be exported from the development environment. Instead, SharePoint developers will need to implement the Lists and Workflows in the deployment environment. The implementation details are included in '3 Design and Implementation'. In contrast, the deployment of the

ASP.NET portion is straightforward. The source code will be included in a .zip file with this document.

The IT Service Request Portal is a proof of concept. There are, at a minimum, twelve other request types that must be implemented. However, a lion's share of the technologies required to implement the system have been researched and used by the development team. JavaScript, ASP.NET, and SharePoint are the main technologies used for system implementation.

In future development, the system must be modified to synchronize with DocuSign and ServiceDesk.

2 SYSTEM ARCHITECTURE

The IT Service Catalog is a self-service portal containing a menu of common Support and System Access Requests (further referred to as “requests”). The system utilizes SharePoint as a backend and ASP.NET as a frontend. All users must authenticate before interacting with the system. Windows authentication will occur before the user reaches the IT Services Portal.

The figure below gives a high level overview of the role ASP.NET and SharePoint play. The frontend controls what views are presented to the current user, depending on the user type. As mentioned earlier, the backend provides a method of storing data and controlling processes. This is achieved through the utilization of lists and workflows; lists store data and workflows control processes.

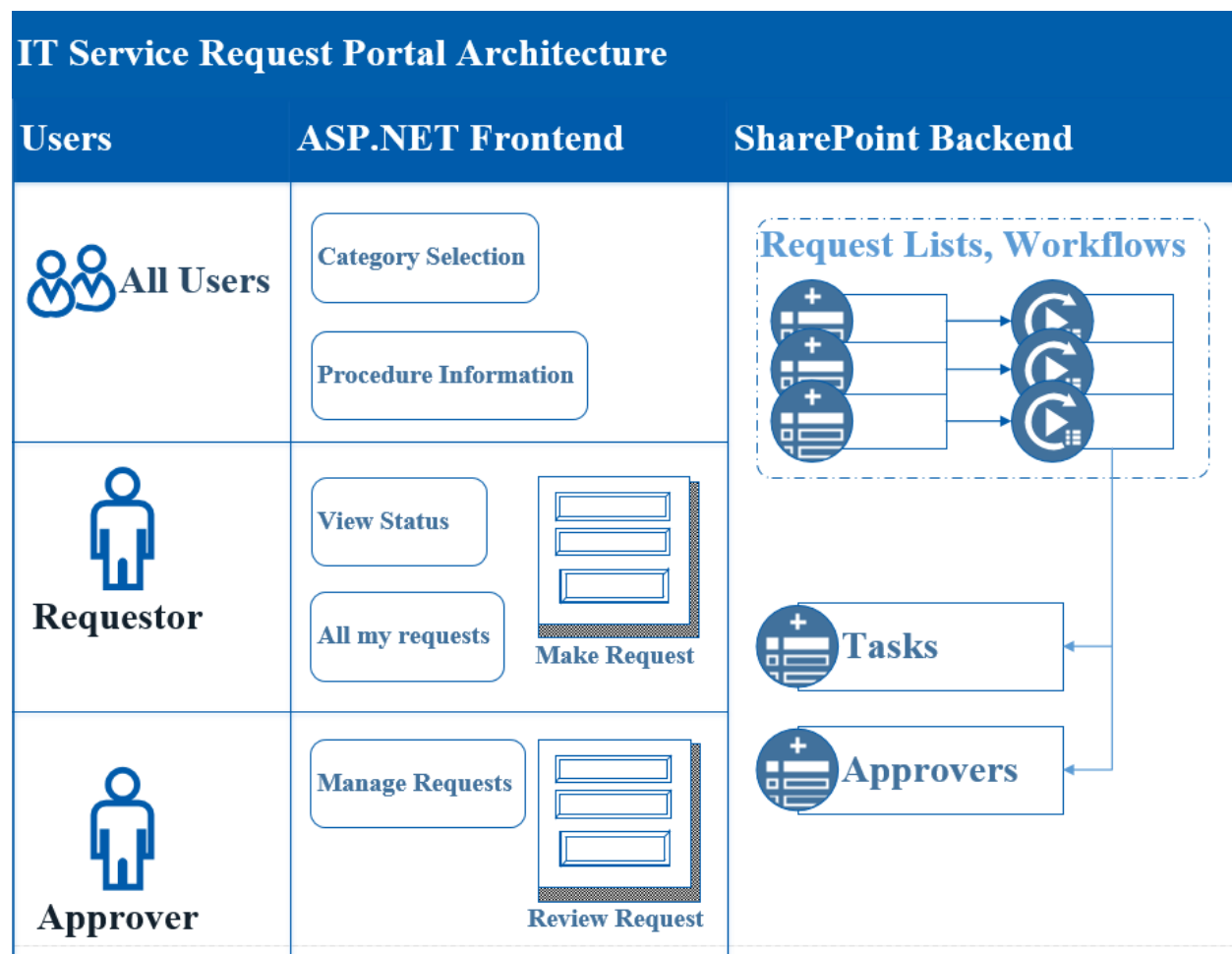


Figure 2.1 IT Service Request Portal Architecture

The system utilizes SharePoint to provide data storage and management. The primary data object is the request instance, backed by a SharePoint item. By SharePoint convention, a

SharePoint item cannot be defined without a related SharePoint list. Each SharePoint list represents a Support and System Access Request type. Each SharePoint list has an associated SharePoint workflow. The workflow contains business logic necessary to process a request through every possible state: pending, submitted, denied, and approved. When new request instances are created the associated item is added to the corresponding list and a workflow instance will start.

The system utilizes ASP.NET pages to provide a user interface. The IT Services Portal and Category Selection pages provides navigation for the user to intuitively find their intended request type. The Procedure Information pages provides information for the user to find answers to frequently asked questions. The Make Request and Review Request forms provide a place to create and modify requests. If a request is created a new SharePoint item is added to its respective SharePoint List. If an existing request is modified, the corresponding SharePoint item is modified and returned to its SharePoint list. The View Status, All my Requests, and Manage Requests provide data management functionality. View Status displays information on one existing request. All My Requests and Manage Requests display information on many existing requests.

For this project, VPN Requests are implemented. Each VPN request is stored as an item in the VPN Requests list. Each time a VPN request item is created, a VPN workflow instance initializes. The workflow will track the request through each approval step and notify requestors and approvers at the appropriate time. The same list-workflow relationship will be copied in future development.

The IT Service Request Portal is implemented with an ASP.NET MVC 4.5 frontend and SharePoint 2010 backend. In general, the ASP.NET frontend controls user experience, while the SharePoint backend stores data and handles process control.

The figure below gives a high level overview of the role ASP.NET and SharePoint play. The frontend controls what views are presented to the current user, depending on the user type. As mentioned earlier, the backend provides a method of storing data and controlling processes. This is achieved through the utilization of lists and workflows; lists store data and workflows control processes.

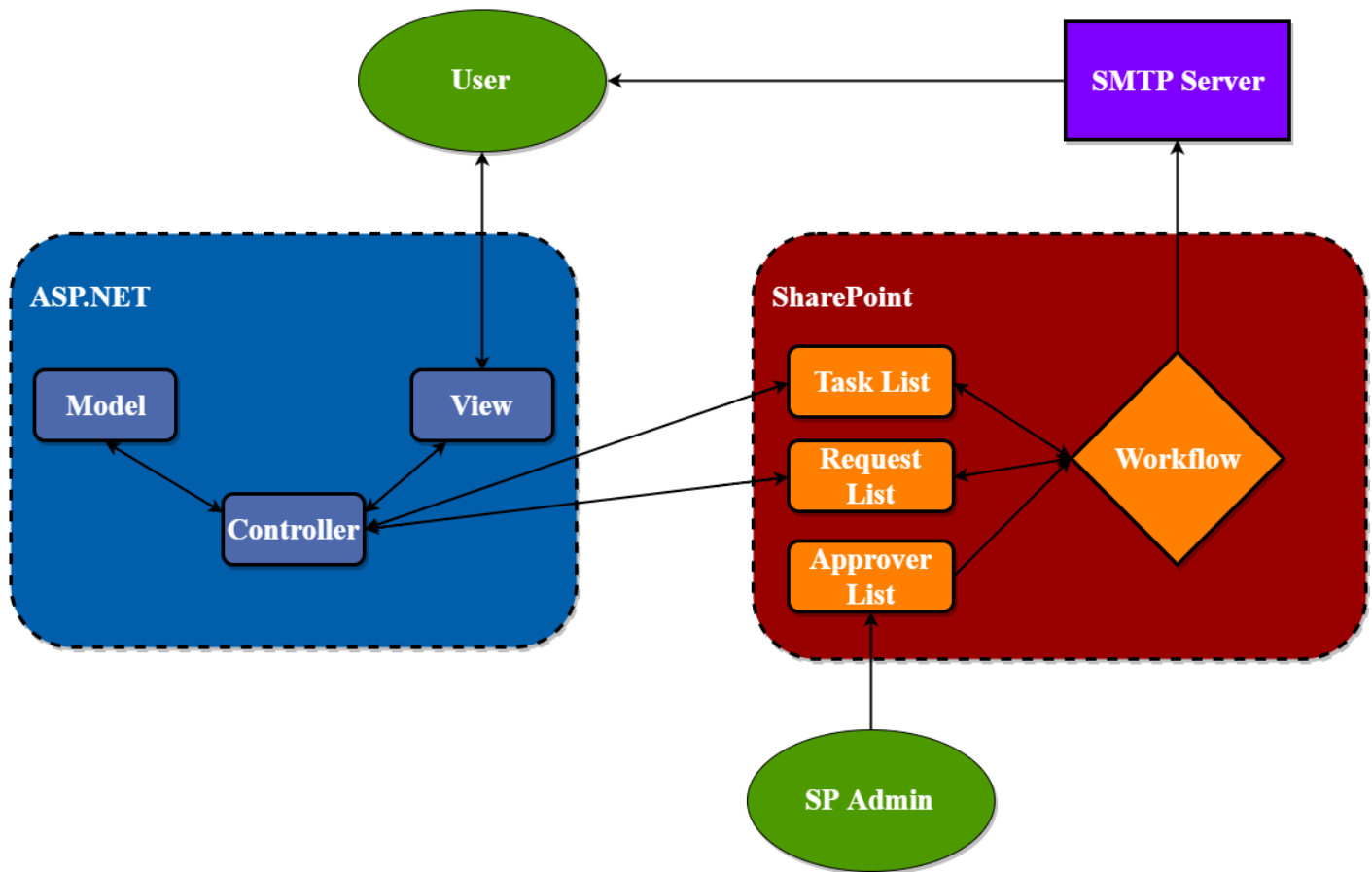


Figure 2.2 System Architecture

This diagram is useful for understanding three main concepts: how our software project was built and organized, understanding that the user will only interact with ASP.NET, and the only backend maintenance required for our solution is that a SharePoint administrator must maintain an up-to-date approver's list.

The development environment contains the following applications:

- Windows Server 2008 R2
- IIS 6
- SharePoint 2010
- SharePoint Designer 2010
- ASP.NET MVC 4.5 or higher
- Visual Studio 2010

The SMTP server is a simple utilization of IIS6.

3 DESIGN AND IMPLEMENTATION

3.1 SHAREPOINT: LISTS AND WORKFLOW

Due to SharePoint 2010 limits, the VPN Request Workflow cannot be exported from the development environment and imported into a deployment environment. Section 4.1 SharePoint: Lists and Workflow will describe the VPN Request List and VPN Request Workflow. The intention of the following sections is to include enough detail such that any SharePoint developer will be able to recreate our steps in the deployment environment.

The SharePoint configuration consists of three lists:

1. VPN Request List used to store information directly related to a request. Each request will have exactly one VPN Request List item associated with it.
2. Approvers List used to store the users who are either a security officer or an IT manager.
3. Task List is a default list that comes with every installment of a SharePoint site. Task items are created in order to assign a user to a request. Each request will have multiple, associated task list items.

The VPN Request Workflow is directly associated with the VPN Request List and is triggered upon a VPN Request List item creation. The workflow will also generate task items in the Task List assigned to a person or group. The workflow uses the Approvers List as a read-only reference. The specifics of each list and the workflows will be listed in the following subsections.

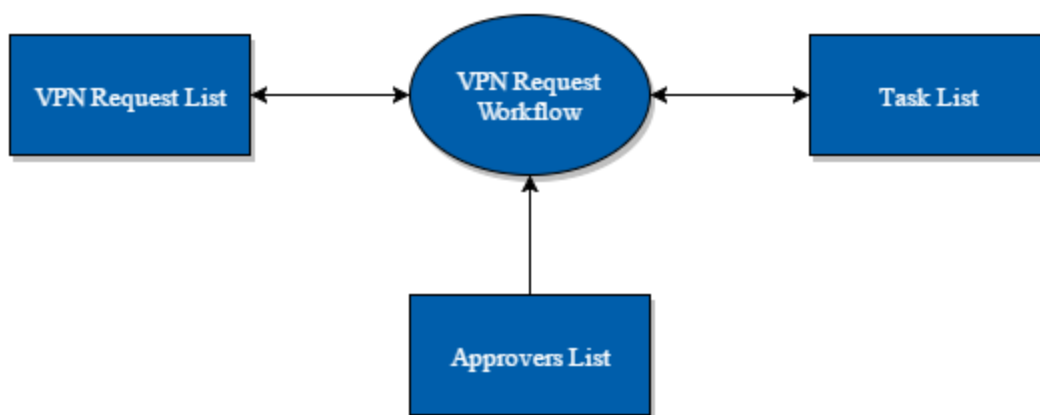


Figure 3.1 SharePoint Architecture

VPN REQUEST LIST

Field Name	Value Type	Additional Details
Agency	Single line of text	
Company Name	Choice	QTC LMCO Other
Company Other	Single line of text	Only filled in if Company Name = Other Enforced in ASP.NET
Machine Owner	Choice	QTC Owned PC Company Asset (Non-QTC) Personal PC
Manager	Person or Group	Manager of the recipient
Office Address	Choice	Diamond Bar, CA - 21700 Copley Dr. Diamond Bar, CA - 1440 Bridgegate Dr. San Antonio, TX - 4400 NW Loop 410 Philadelphia, PA - 1617 JRK. Blvd. Though these are possible choices, we can still set Office Address to

		the fill in choice if Office Location != QTC Admin Office though ASP.NET
Office Location	Choice	QTC Admin Office QTC Clinic Other
Radius Profile Other	Single line of text	Fill in if Radius Profile Select = Other
Radius Profile Select	choice	Full Access QTC Web Access R CRM Contractor R Indexer Remote R LMCO Support R Neudesic Contractor R QA remote R SP Portal Contractor R Telehealth REVPN QTC Transcribers Other
Reviewer Comments	Multiple lines of text	
Systems List	Multiple lines	

	of text	
Telephone Extension	Number	
VPN Access End	Date and Time	
VPN Access Start	Date and Time	
VPN Justification	Multiple lines of text	50-100 words Enforced in ASP.NET
VPN Profile Select	Choice	Checkboxes: QTC, Transcriber, Other
VPN Recipient Email	Single line of text	Email address format Enforced in ASP.NET
VPN Recipient First	Single line of text	
VPN Recipient Last	Single line of text	
VPN Request ID	Single line of text	

VPN Request Status	Choice	Pending Manager Approval Pending Security Officer Approval Pending IT Manager Approval Approved Rejected by Manager Rejected by Security Officer Rejected by IT Manager
VPN User Code	Number	3 digits Enforced in ASP.NET
VPN User Dept	Choice	CLS CRP IT STS VAS VHA
VPN User Status	Choice	QTC Regular Employee Temp Employee Contractor Consultant Transcriber

		QTC Provider LMCO Employee (Non-QTC)
Work Phone	Single line of text	### - ##### format Enforced in ASP.NET
VPN Requester	Person or group	

APPROVERS LIST

Field Name	Value Type	Additional Details
User	Person or Group	Current supports 1 security manager and 1 IT manager
Title	Choice	Security Manager/IT Manager

TASK LIST

As mentioned in previous sections, our task list is an out-of-the-box implementation, but we modified certain fields as a workaround solution to some of SharePoint's limitations. The fields listed below are used in our solution; the default task list comes with many unused fields.

Field Name	Value Type	Additional Details
ID	Number	Built in ID No configuration required

Title	Single line of text	More information below
Assigned to	Person or group	The person the task is assigned to
Outcome	Choice	Approved Rejected

The title field was modified to serve as our makeshift identifier. The format we created is: Task Type/Step/Request ID. This allows for easy-to-extend ASP.NET controller code in the future since all request types can share the same task list.

WORKFLOW

Our solution implemented a list workflow with 3 approval workflows in between. This section will go through each of the five steps on the specifics of our workflow. I will refer to fields and their lists as list.field.

Step 1: Setting VPN Request ID: Our corporate liaisons requested that request IDs be in the thousands place (I.E. 10005). Our very first step is to set the VPN Request List Request ID to VPN Request List.ID + 1000. Note that Request ID is not stored as a number because floating point values will be displayed.

Step 2: Setting email variables: The next step is to set two static variables. Review Request URL will be set to the URL of the ASP.NET review page for a specific review. My Request URL will be set to the URL of the ASP.NET My Request page for a specific request.

Examples:

Review Request URL = <http://qtcserverhostname/portal/reviewrequest/10005>

My Request URL = <http://qtcserverhostname/portal/MyRequest/10005>

Step 3: Manager Approval: At the start VPNRequestList.VPNRequestStatus is set to “Pending Manager Approval”. Then an approval process is started with VPN Request Status.Manager.

The behavior of this single approval process is modified to send out emails in the QTC format (the process already has emails notifications built in). Use the URL variables to link the user to the correct pages.

After the approval process we check the built in variable `IsItemApproved`. If it equals no, then we set the `VPN Request List.Request Status` to “Rejected by Manager”, set the workflow status to “Rejected”, and stop the workflow and log “Rejected at Manager Approval Step”.

Step 4: Security Manager Approval: This step is very similar to step 3. To start, we set the `VPN Request List.VPN Request Status` to “Pending Security Manager Approval”. Then we start another approval process, but this time we will reference the `Approvers List`. Start the approval process with `Approvers List.Users where Approvers List.Title = IT Manager`.

Again, make sure to set the emails to the correct template or format.

After the approval process we check `IsItemApproved1`. If it equals no, we set the `VPN Request List.VPN Request Status` to “Rejected by Security Manager”, workflow status to “Rejected”, and stop the workflow and log “Rejected at Security Manager Step”.

Step 5: IT Manager Approval: At the start of the final step we set the `VPN Request List.VPN Request Status` to “Pending IT Manager Approval” before starting the final approval process with `Approvers List.Users where Approvers List.Title = IT Manager`.

Make sure to set the email to correct template or format.

After the approval process we check `IsItemApproved2`. If it equals no, set `VPN Request List.VPN Request Status` to “Rejected by IT Manager”, set workflow status to “Rejected”, and stop the workflow and log “Rejected at IT Manager Approval”.

If `IsItemApproved2` equals yes, set `VPN Request List.VPN Request Status` to “Approved”, set the workflow status to “Approved”, and stop the workflow and log “Approved”.

3.2 IT SERVICE PORTAL USER INTERFACE

The views of the IT service request portal are built on top of the razor view engine. HTML5, CSS3, and Bootstrap, are used to display all of the views in the portal. JQuery is used for a frontend scripting language mostly in form validation and some in portal design.

The UI was built to implement the following site flow:

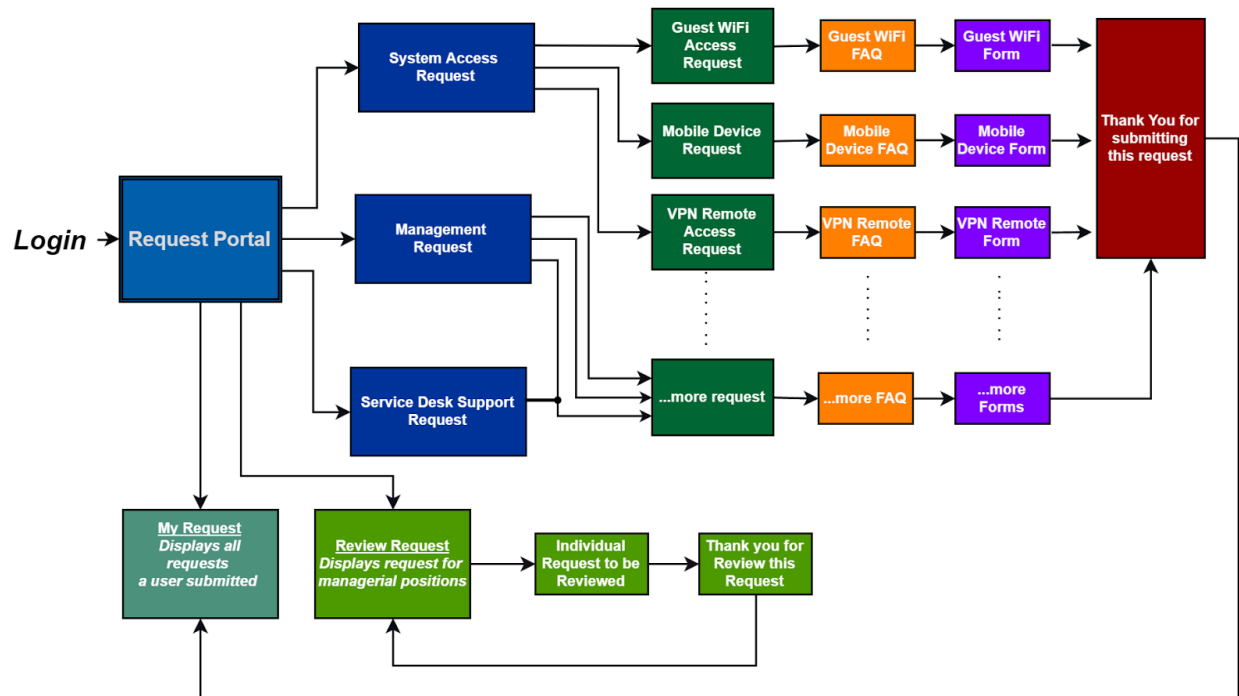


Figure 3.2 Site flow map

The site flow begins at the landing page or “request portal”. From here the user may navigate to three areas, grouped by a common functionality: tracking requests, reviewing requests, and submitting requests.

MY REQUEST PAGE

The My Request pages allows users to track requests they have submitted. The main page displays a list of all the current user’s requests. The page also comes with a handy filter that allows the user to filter the list by statuses (approved, rejected, or pending). In the list view, the user is not shown a lot of detail, but it’s enough detail to aid the user in finding a request.

My Requests			
Pending	Approved	Rejected	All Requests
Request Type	Request ID	Date Submitted	Status
VPN Access	1027	04/30/2016	Pending Manager Approval
VPN Access	1037	05/04/2016	Approved
VPN Access	1038	05/05/2016	Pending Manager Approval
VPN Access	1039	05/05/2016	Approved
VPN Access	1040	05/05/2016	Rejected by Manager
VPN Access	1041	05/05/2016	Rejected by Manager
VPN Access	1047	05/10/2016	Pending Manager Approval
VPN Access	1048	05/10/2016	Pending Manager Approval
VPN Access	1049	05/10/2016	Pending Manager Approval
VPN Access	1050	05/10/2016	Approved

Figure 3.3 Screenshot of My Request list view

Upon selecting a request, the user is show more detailed information regarding the selected request.

REVIEW REQUEST PAGES

The review requests list view UI is similar, but it displays all requests assigned to the user. This page also enables the user to filter the page by the current status of the request. Upon selecting a pending request assigned to the user, the user is shown a form that prompts his/her approval. Upon selecting a request that was once reviewed by the user (this can be in any status), the user is shown a view-only page of all the details pertaining to that request.

SUBMITTING NEW REQUESTS

To submit new requests, the user must first select the type of request they wish to submit. The requests are broken down into three groups. The user is brought to an FAQ page and then a form. The input elements on the form contain client-side validation consistent with the details of the specific request type.

For the VPN Request Form, most validation simply requires a specific number of alphanumeric characters (Figure 3.4). There is also more complex validation which updates the form based upon data entered into specific input elements (Figure 3.5). The VPN Request Form has client side validation built into every input element. All validation is written in JQuery. The validation code can be found on the ASP.NET web application.

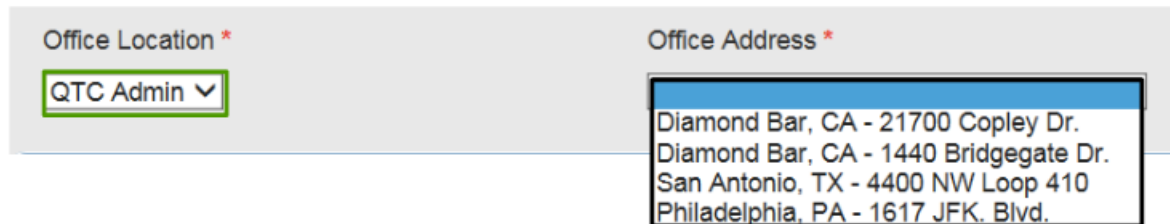


First Name *

J

Length must be between 2 to 16 characters.

Figure 3.4 Simple validation



Office Location *

QTC Admin

Office Address *

Diamond Bar, CA - 21700 Copley Dr.
Diamond Bar, CA - 1440 Bridgegate Dr.
San Antonio, TX - 4400 NW Loop 410
Philadelphia, PA - 1617 JFK Blvd.

Figure 3.5 Complex validation

The purpose of the validation is to ensure the requestor provides all required information. That eventually saves time for everyone involved because the validation eliminates all chances of invalid user input.

USER INTERFACE VIDEOS

These videos have been created by the development team. The basic functions of the IT Service Catalog are demonstrated in these videos.

My Request: <https://www.youtube.com/watch?v=h15R-q73zJQ>

Request Portal: <https://www.youtube.com/watch?v=ikcw6KAHRE0>

VPN Remote Form: <https://www.youtube.com/watch?v=R8iiwUmijSA>

3.3 CONNECTING ASP.NET AND SHAREPOINT

ASP.NET MODEL CLASSES

The web solutions model classes are kept under the default Models folder. The contents of that folder contain:

- Internal Server Names (Folder)
- VpnRequest.cs (Model Class)
- SpConnectionVPN.cs (Client/Driver)

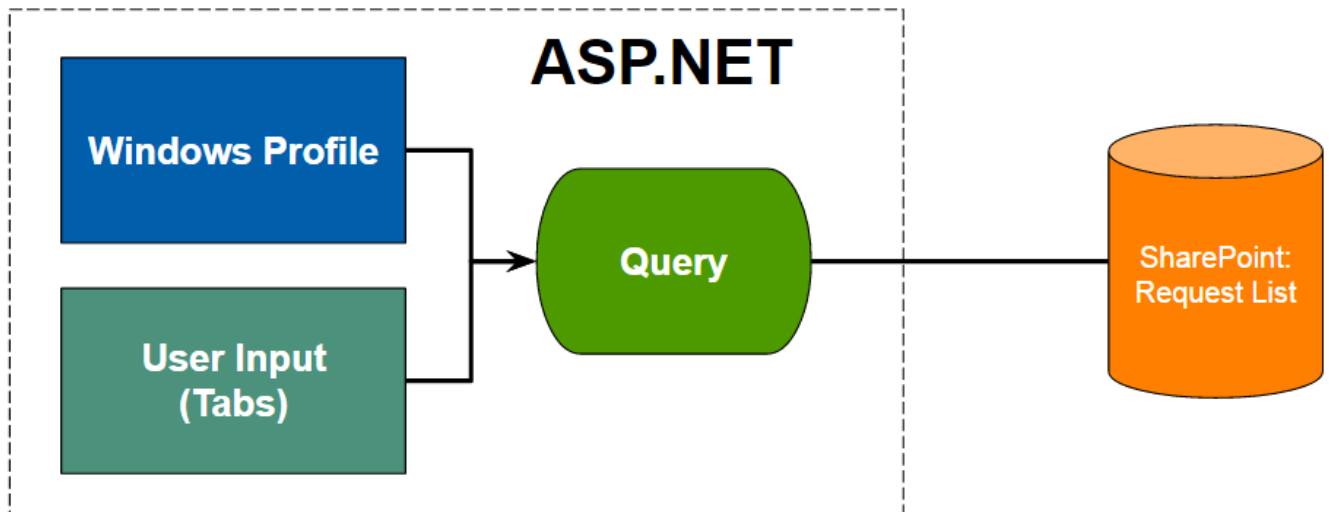
The Internal Server Names folder contains static variables that hold the internal SharePoint server name for fields. For example, InternalVpnRequestNames.internalVpnRecipientFirst is a variable that holds the internal server name for the field 'VPN Recipient First Name' in the VPN Request List. These files need to be manually updated and maintained as the code base expands.

The VpnRequest.cs model class is a C# model class that represents all the details of a request. The VpnRequest.cs model is used solely on the ASP.NET project, data from the SharePoint server is mapped to this class. This model is also sent to the project views.

The SpConnectionVPN.cs class is not strictly a data model class. Rather, it is a driver or client that communicates with the SharePoint server. SpConnectionVPN uses the Client-Side Object Model (CSOM) to create, read, and update SharePoint list objects.

To create or update CSOM data objects, the data must be mapped from a VpnRequest.cs model to a CSOM list object. After that the CSOM's Update() method is invoked to commit these changes to the SharePoint site.

Reading CSOM data objects can be complicated. Our project uses the CSOM's Extensible Markup Language (XML) querying capabilities to specify exactly what data to extract. The current user should only see a subset of all the VPN Requests on the list. Our XML queries also use the current user's windows profile (detected through Windows Authentication) to determine what type of data should be retrieved. The figure below depicts a high level overview of how the MyRequests query is built. Note, this will return all the requests that should be displayed on the current user's Myrequests tab.



Windows Profile = Request List.Requester
AND
User Input (Tabs) = Request List.Status

Figure 3.6 Example CSOM XML Query, MyRequests Query

CONTROLLERS

The controller classes and methods follow the standard convention for ASP.NET MVC. If a controller class is named `FooController.cs` with an action method `Bar()`, the url will be `Foo/Bar`, and there will be a folder named `Foo` in the Views folder with a file named `bar.cshtml`. By default `Bar()` is linked to the `bar.cshtml` view file.

Due to this convention, it's very easy to find a codepath so there is no need to define and describe each controller and its method. The controller mechanics are relatively straightforward as well. Anytime the application needs data from the SharePoint lists, the driver class `SpConnectionVPN.cs` is invoked.

Occasionally, a controller method will contain a few lines of code to convert data types. This is due to a division of labour (separate people worked on SharePoint, views, and controller methods), confusion, ambiguity, and sloppiness. Ideally, correct data types should be correct already when they come in to the controller

3.4 RUNNING THE ASP.NET APPLICATION

To run the ASP.NET web application, you will need to first spot check the internal names for the list fields. Most of them should not need to be changed, but some will. You must find a way to find the internal names of fields; there are multiple methods. The easiest is to use Visual Studio 2010 Professional (or higher) server explorer.

After you are able to find the internal server names for fields, find the Models folder in the solution explorer in Visual Studio. Under the Models folder, navigate to Internal Server Names.

The first file is named InternalTaskNames.cs. These are all the internal names related to the Task List on SharePoint. The variables are named with a standard convention: 'internalTask' followed by the field name on SharePoint. Using your method of finding internal server names, check that they are correct.

The second file is named InternalVpnRequestNames.cs. These are all internal names related to the VPN Request List and the Approvers List on SharePoint. The code is commented and the variable names follow a consistent convention. Again, make sure the internal server names match up. Also, make sure the SiteUrl, ListName, and ApproversListName are variables are correct. SiteUrl should be the URL of the team site.

After these model classes are correctly configured, the project will run.

4 CONCLUSIONS

Our project isn't perfect and there's clearly a lot left to develop, but we are proud of our product. We are proud of the lows we overcame and the highs we achieved. Because of the project we were able to learn technical skills we otherwise would not have learned in our school curriculum. We were able to become familiar with developing robust workflows and lists in SharePoint. We were able to create aesthetically and functionally pleasing UI with ASP.NET. Combined, we were able to create a sleek, elegant solution to a complex business problem.

Beyond the concrete technical skills, we were able to experience what it meant to solve a problem with software. We learned it took time, due diligence, patience, and persistence to develop a smooth, fine-tuned piece of software. We learned to embrace the bumps in the road, the problems, and the unexpected issues as intrinsic elements of software development.

The IT Service Request Portal, as it stands, is a minimum viable product. There are still many other request types that must be implemented. DocuSign and ServiceDesk integration must still be added. The IT Service Request Portal requires a lot more development time.

However, many of the tools required to develop the IT Service Request Portal have been researched and are well understood by the development team. If the development team were allowed to continue work, development time would go by quickly. Unfortunately, due to the nature of a senior design project, the development team must disband and the project will be handed off to the client. We, as the development team, hope that we have provided enough information to continue development. We also hope this software system brings value to our client and helps to solve their business problem in a meaningful, lasting way.