Software Architecture for Paychex Out of Office Application

Version 2.3

Prepared by:
Ian Dann
Daquanne Dwight
Tom Eiffert
Elysia Haight

Rochester Institute of Technology
Paychex

March 10, 2013

Revision History

Version	Revision Date	Changes Made	Justification	Authors

1.0	1/15/13	Initial Revision	Creation of template	Elysia Haight
1.1	1/15/13	Added and completed Technologies and Justifications Section	Final decision on technologies made	Ian Dann Daquanne Dwight Tom Eiffert Elysia Haight
2.0	1/22/13	Inserted view diagrams, completed missing sections	Completed second draft	Ian Dann Daquanne Dwight Tom Eiffert Elysia Haight
2.1	1/27/13	Updated data view	Updated data view	Daquanne Dwight
2.2	1/31/13	Updated data view	Updated data view	Daquanne Dwight
2.3	3/10/13	Updated data view	Updated data view	Daquanne Dwight

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Overview

This document is organized to first detail technologies utilized in the development of the software, along with reasons and justifications. The goals of the system are then listed, followed by the architectural views. Finally, definitions and references are available.

2. Technologies and Justifications

2.1 Programming Language

2.1.1 C#

C# is a very powerful language. Able to be utilized for standalone desktop applications as well as server-side functionality for web applications, C# is quickly becoming the go-to language for most large companies. It is strongly supported and flexible, allowing for simple library inclusion through the usage of DLLs.

C# will be used in the Paychex Out of Office Application to implement server-side functionality of the system. This will allow for use of plugins to access our database (see Section 2.2), LDAP (see Section 2.4), and PDF report generators (see Section 2.3).

2.1.2 ASP.net

ASP.net is a server-side web application framework that produces dynamic web pages. It combines static HTML markup and dynamic definitions of server-side Web/User Controls. When paired with C#, complex web application development is greatly simplified.

The MVC Framework will be utilized to enforce the Model View Controller architectural pattern used in this project.

2.1.3 JavaScript (jQuery)

jQuery, an open-source library built on JavaScript, will be used to implement client-side functionality in the project. jQuery offers all the same functionality as JavaScript, with simplified implementation.

2.2 Database

2.2.1 MySQL

MySQL is a relational database provided by Oracle. MySQL will be used as the database for all backend storage. MySQL is free/open source, has a large community, and is well documented.

2.2.2 MySQL Connector/.Net

MySQL Connector are drivers provided by Oracle to connect .Net applications to a MySQL database.

2.3 PDF Reporting Tools

2.3.1 PDFsharp

PDF sharp provides an easy solution for processing PDF documents at runtime. It is a .NET library that is available to use with any of the .NET languages and is open source with MIT licensing. It's object model easy enough to understand yet powerful enough to create PDF's from any text source.

2.4 Email Integration

2.5 LDAP

2.5.1 LDAP C# Tie-In

A C# Library created to allow of easier interfacing with LDAP systems. This will be utilized to connect to existing Paychex systems with minimal difficulty.

2.6 Scheduler

2.6.1 Quartz.NET

Quartz.NET is a job scheduler for .NET framework. It allows jobs to be repeatedly executed based off time of day and day of the week.

3. Architectural Goals and Constraints

The main goal of the Paychex Out of Office is to achieve usability. This includes the system's ease of use, ease of learnability, and pleasant aesthetics. The system is to be optimized for mobile use, with computer use as a secondary concern.

The system is also intended to be reliable; system downtime is to be minimal, and scheduled tasks should operate at the designated times.

4. Architectural Representation

The architectural representation enumerates the views that will describe and visualize the major components of the system and how they interact. Each view is described, displayed, then components are explained for ease of understanding. These views are high-level and generalized, and are not a solid representation of the specific system design.

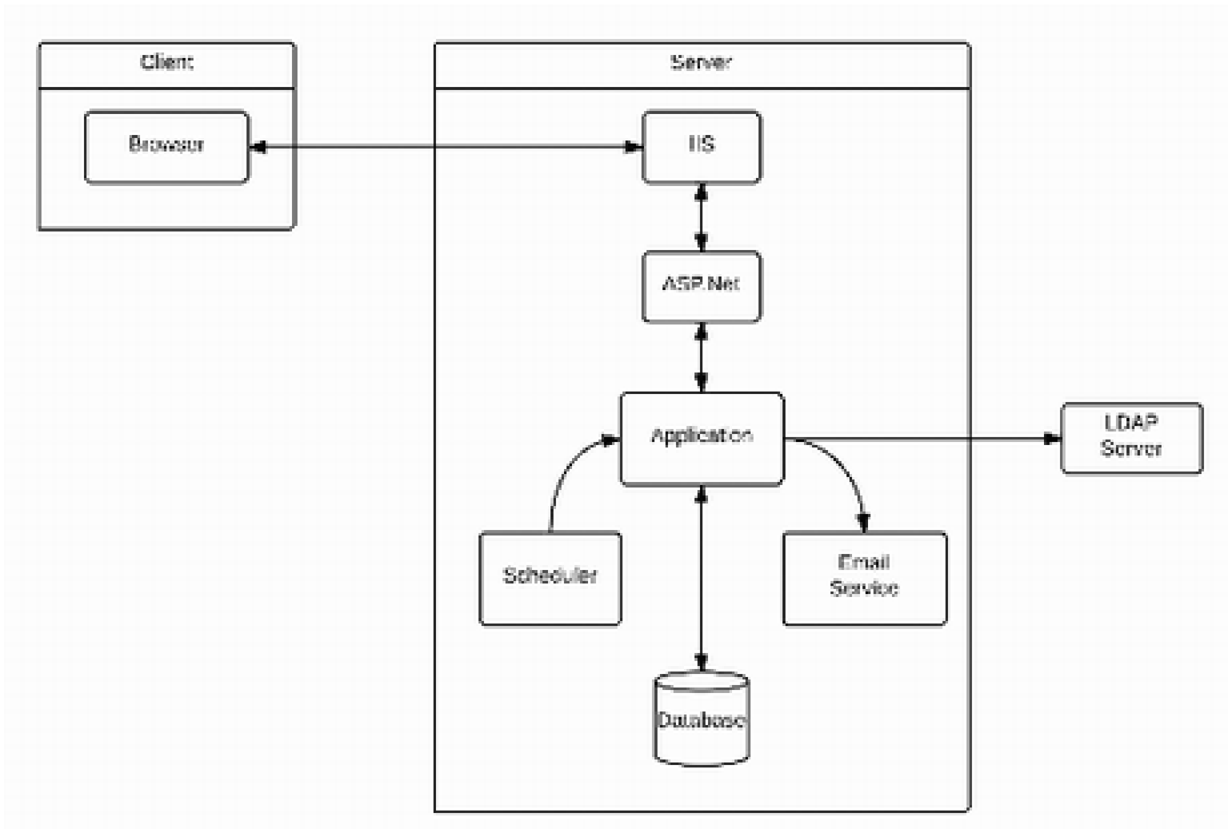
Section 5 Deployment View, Section 6 Layer View, and Section 7 Layer View comprise the system's architectural representation.

5. Deployment View

5.1 Overview

The Deployment View shows what is necessary for system execution. The system-external software and hardware components are listed to ensure a valid setup has been created for testing and deployment purposes.

5.2 Deployment Architecture



5.2.1 Client

The following operating systems are to be supported by the system: iOS 4+ and Android 4.0+. The client browser running the application requires Javascript.

5.2.2 Server

The server operating system is to be Windows Server 2008 R2. The server must have the following applications/services (and minimum versions) installed to perform:

IIS: 7.5
ASP.NET: 4.0
Scheduler: Quartz.NET 2.1.2
Email Service: Papercut
Database: MySQL

5.2.3 LDAP Server

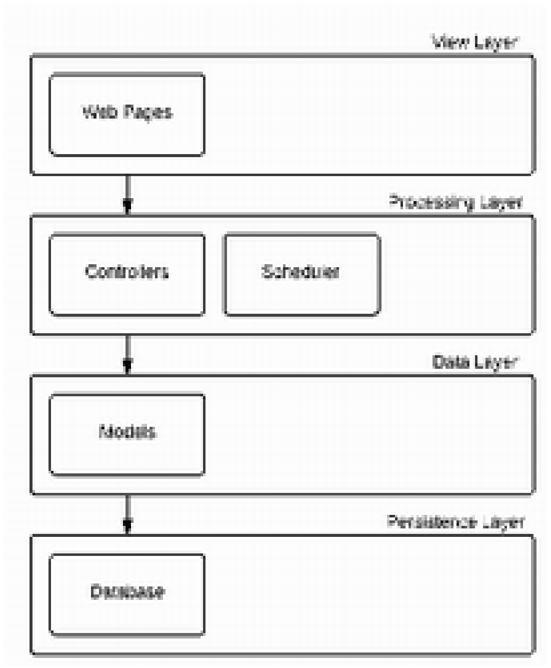
TBD

6. Layer View

6.1 Overview

The layer view displays the separation of concerns in terms of the system's modules, and the flow of communication between the layers. Our system is designed to be linear, such that requests from the highest layers are handled by the layer immediately below it.

6.2 Layers



6.2.1 View Layer

The View Layer contains the implementation of the web pages. It interacts with the controller layer to display relevant data on the page.

6.2.2 Processing Layer

The Processing Layer contains the controllers (utilized for the MVC architecture) and the Scheduler. The controller classes handle communications between the Data and View Layers, providing a separation of concerns. The scheduler communicates with the Data Layer to perform its designated task(s).

6.2.3 Data Layer

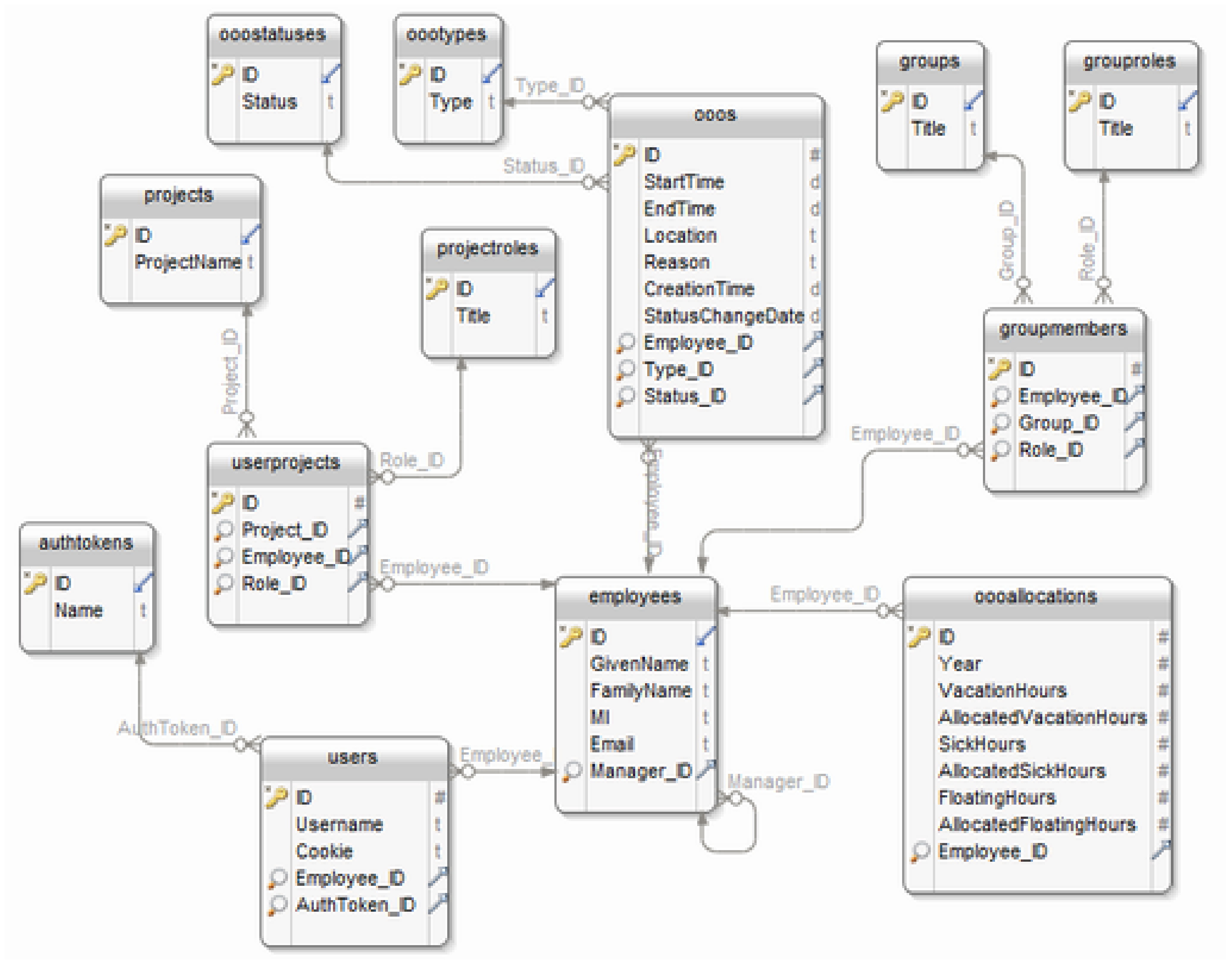
The Data Layer stores web page information during run time, and retrieves/organizes data retrieved from the Persistence Layer.

6.2.4 Persistence Layer

The Persistence Layer contains any persistent storage media. It currently contains the database, which is accessed by the Model Layer.

7. Data View

Below is the database schema representing how data is to be organized in a relational database:



8. Quality

The system’s usability is supported in the architecture by keeping a valid separation of concerns between layers. This is intended to aide system learnability. The user interface is also designed to be as consistent between pages as possible.

9. Definitions, Acronyms, and Abbreviations

- DLL - Dynamic Link Library; Microsoft’s implementation of linking libraries that may contain any combination of code, data, or resources.
- MVC - Model, View Controller; An architectural design separating the background work from the user interface

10. References