Performance @ppraisal: Using Empirical Data to Evaluate Job Performance

José A. Aponte-Lucena
Computer Engineering
Eduardo Sobrino, Ph.D.
Electrical Engineering Department
Polytechnic University of Puerto Rico

Abstract *— The Performance @ppraisal is a three-tier web application implemented in the U. S. Probation Office for the District of Puerto Rico to evaluate the job performance of their staff using empirical data extracted from external data sources. It was implemented using the Model-View-Controller (MVC) design pattern in PHP. The main objective is to present a robust software design, and an implementation with source code easy to understand and maintain. This publication is presented as a Project in Internet Engineering of the Polytechnic University of Puerto Rico.*

Key Terms *— Appraisal, Controller, Evaluation, Model, MVC, MySQL, Performance, PHP, View.*

INTRODUCTION

The United States Probation Office for the District of Puerto Rico requested a web application to develop and apply policies and procedures to evaluate their employees' performance and to plan their professional development. These procedures must be applied consistently through the agency.

Online documentation will present guidelines and best practices as guidance to the supervisor to create the evaluation instrument, to assist them when evaluating their employee's job performance, and to recommend a development plan based on the evaluation results. In addition, employees can consult this documentation when reviewing their performance evaluations.

The objective of this publication is to present technical details about the software design and implementation of this web application. In particular, we focus in all software design principles used to create a robust web application, easy to maintain, and applying the most up-to-date technology to optimize software longevity.

PERFORMANCE @PPRAISAL

Performance @ppraisal is a three-tier web application implemented in the U. S. Probation Office for the District of Puerto Rico to evaluate the job performance of their staff. The implementation of this application is presented as a project in Internet Engineering for the Polytechnic University of Puerto Rico.

Empirical data obtained from external data sources, such as: case management reports [1], time and attendance records, and training history, is preferred over more subjective criteria when evaluating an employee's job performance. Finally, the appraisal ratings should be assigned in a standard manner through the agency, regardless of the supervisor scoring the evaluation questions.

The three-tier design was implemented using the Model-View-Controller (MVC) design pattern to separate the user interface (view) from the business logic (controller) and the database operations (model).

DESIGN AREAS

The first step of the software design process was to identify the use cases for the Performance @ppraisal application. Use Case specifications were grouped by design area based on their functionality.

To make the software design and source code easier to understand for future maintenance, all object classes, including controllers and views, were grouped in packages a directory structure corresponding to the following design areas used to

classify the use case specifications. See Figure 1 below for a Design Area Diagram.

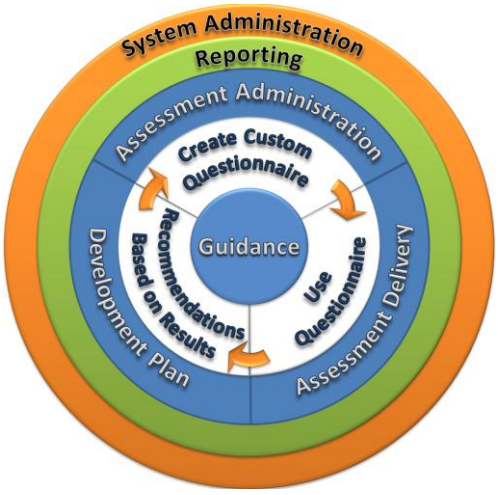

Figure 1
Design Areas Diagram

- **System Administration:** Includes modules for the System Administrator to maintain global configuration parameters for the Performance @ppraisal web application. In particular, the implementation of role-based security, to restrict system access to authorized users only and to assign permission rights.
- **Reporting:** Contains modules to generate reports to measure statistics, to meet deadlines, and to obtain historical data of the Performance @ppraisal web application.
- **Assessment Administration:** Comprises all the modules to create the assessment instrument – custom performance evaluation questionnaires – and its components.
- **Assessment Delivery:** Stores modules for supervisors to use the assessment instrument to evaluate the job performance of their staff. It also includes modules for employees to review their performance evaluations.
- **Development Plan:** Contains modules for supervisors to create a development plan for their employees based on the results of their job performance evaluation.
- **Guidance:** Includes modules to provide online documentation about guidelines and best practices to create the assessment instrument. In addition, it provides documentation and minimal criteria to the supervisor on how to use the assessment instrument, and to the employee to help them understand the criteria used to evaluate their job performance. Finally, it also provides recommendations for the creation of a development plan based on the job evaluation results.

See Figure 1 below for a Design Area Diagram.

USE CASES

This section includes a detailed description of each use case grouped by design area.

System Administration

The following use case specifications implements the role-based system security:

- **To Create a User Account:** The System Administrator creates a user account with login credentials (User Id and Password), and assigns it one of the following role-based permission levels to access the Performance @ppraisal web application: System Administrator, Supervisor, or Employee.
- **To Login:** The Performance @ppraisal System validates the login credentials against the database. When the validation succeeds, it creates a global session variable with the User's Name and Security-Role Permission. After three failed login attempts, the Performance @ppraisal system locks the user account to avoid an intruder to guess the login credentials of an authorized user.
- **To Receive Login Information By E-mail**: Authorized users can request a copy of their login credentials to be sent in an e-mail message when they forget this information.

Reporting

- **To Print a Report:** The Supervisor can print reports from the Performance @ppraisal system to measure statistics, meet deadlines, and obtain historical data. Job performance

statistics will help him / her to detect weak areas in the staff performance in order to provide remedial action such as additional training. In a similar way, management can identify employees with outstanding performance evaluations in a particular area that can help to train other staff in that area of expertise. Deadline reports are useful to the supervisor to schedule the job performance reviews on time and to follow up on the development plan actions.

Assessment Administration

The following use case specifications describe how to create the aassessment instrument, the performance evaluation questionnaire:

- **To Create a Performance Appraisal Area:** All questions in a job performance evaluation are grouped by appraisal area to make the assessment instrument easier to understand. In addition, by grouping the scores of questions in the same appraisal area, the supervisor can identify those areas where the employee performance needs improvement.
- **To Obtain Empirical Data from an External Data Source:** Web services are used to extract empirical data from external data sources to be used in the job performance evaluation, such as: case management reports, time and attendance records, and training history. The web service connection parameters are stored in a database table. The records in this table can be linked to one or more evaluation questions. For consistency and ease of implementation, all web services require the same three (3) input parameters: Staff ID, From Date, and To Date.
- **To Create a Performance Evaluation Question:** The Supervisor creates an evaluation question to include in the assessment instrument. This question will belong to one of the existing appraisal areas.
- **To Create a Performance Evaluation Questionnaire:** The assessment instrument used to evaluate job performance is a custom questionnaire. It is composed of questions in different appraisal areas. The questions vary according to the job description and duties assigned. A Supervisor can create custom job performance evaluation questionnaires, and include only those questions pertinent to the job description of the Employee to be evaluated. Additionally, the supervisor can modify the criteria used to score a question as Outstanding, Satisfactory, Marginal, or Unacceptable.

Assessment Delivery

This section includes use case specifications related to the use of the assessment instrument:

- **To Evaluate an Employee's Job Performance:** Each evaluation question is rated with a numeric score as shown in Table 1 below. The average of all individual scores is used to appraise the overall employee's job performance. The Performance @ppraisal application will display detailed information of the criteria used to score each evaluation question.

Table 1
Evaluation Question Scores

Description	Numeric Value
Outstanding	3
Satisfactory	2
Marginal	1
Unacceptable	0

- **To Review an Employee's Job Performance:** Employees receive an automated e-mail notification when their performance evaluation is ready for review. They can examine the job evaluation online, with all the external empirical data used to score the questions. Employees can add comments to any of the question scores. In addition, they can add a comment to the overall job performance evaluation. These comments are stored in the database, and an e-mail notification is sent to the supervisor indicating that the job performance evaluation was reviewed by the employee.

Development Plan

The following use case for development plans based on the results of the job performance evaluation:

- **To Create a Development Plan:** Based on the performance evaluation results, supervisors can create a development plan to assist their employees to strengthen existing skills and to acquire new ones. It will also set goals with specific deadlines to measure developmental progress. The Performance @ppraisal system will suggest some remedial action to be included in the development plan based on the job performance evaluation results.

Guidance

The following use cases to provide online documentation with guidelines [2] and best practices for supervisors to create and use custom assessment instruments:

- **To Provide Guidance and Best Practices to Create an Evaluation Questionnaire:** The Supervisor consults the Performance @ppraisal Evaluation Knowledgebase to read online documentation about the performance evaluation policies. In addition, he / she can read articles about best practices to evaluate an employee's job performance. The user can browse documents by area or search by title or description.
- **To Provide Guidance and Minimal Criteria to Assign Scores:** The Supervisor can consult in the Performance @ppraisal Evaluation Knowledgebase articles about best practices when evaluating an employee's job performance. The user can browse documents by area or search by title or description.

DATABASE DESIGN

The database design was based on the software requirements specifications (SRS) extracted from the use case specifications.

See the Entity-Relationship (ER) Diagram for Assessment Administration in Figure 2 below.

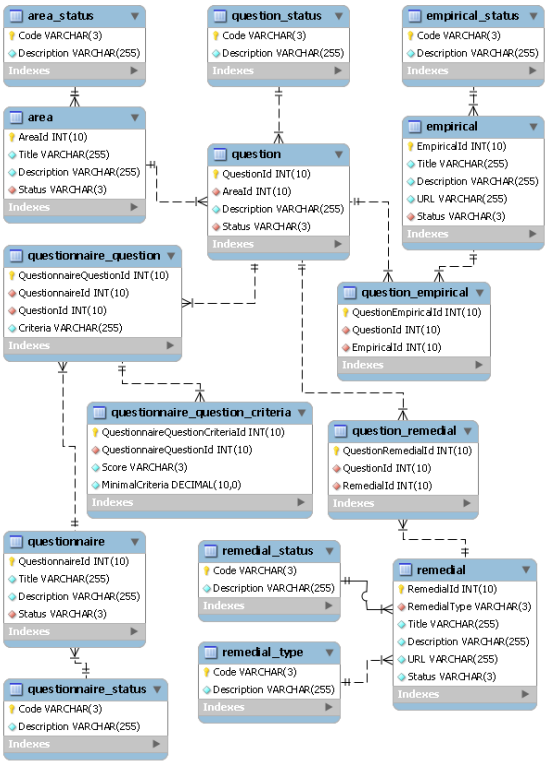

Figure 2

Entity – Relationship Diagram: Assessment Administration

See the Entity-Relationship (ER) Diagram for Assessment Delivery in Figure 3 below.

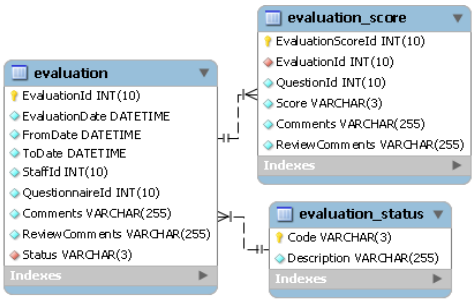

Figure 3

Entity – Relationship Diagram: Assessment Delivery

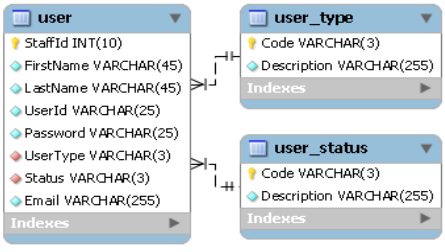

Figure 4

Entity – Relationship Diagram: System Administration

Stored Procedures

To make the source code easier to understand and to maintain, and to optimize the system performance, all database operations were encapsulated in stored procedures. See a list of stored procedures in Table 2 below.

Table 2
Stored Procedures: Assessment Administration – Evaluation Questions (EQ)

Name	Description
sp_area_create	Create a new Appraisal Area
sp_area_delete	Delete an Appraisal Area
sp_area_read	Read an Appraisal Area record
sp_area_update	Update an Appraisal Area
sp_question_available _empirical_grid	Data grid of Empirical Data Sources not related to an EQ
sp_question_available _remedial_grid	Data grid of suggested Remedial Actions not related to an EQ
sp_question_create	Create an EQ
sp_question_criteria	Read the minimal Criteria to score an EQ
sp_question_delete	Delete an EQ
sp_question_ empirical_create	Create a new related Empirical Data Source for an EQ
sp_question_ empirical_delete	Delete a related Empirical Data Source from an EQ
sp_question_ empirical_grid	Data grid of Empirical Data Sources for an EQ
sp_question_ empirical_read	Read a record of the Empirical Data Sources related to an EQ
sp_question_ empirical_update	Update the Empirical Data Source of an EQ
sp_question_read	Read an EQ record
sp_question_ remedial_create	Create a new suggested Remedial Action for an EQ
sp_question_ remedial_delete	Delete a suggested Remedial Action for an EQ
sp_question_ remedial_grid	Data grid of suggested Remedial Actions for an EQ
sp_question_ remedial_read	Read a record of suggested Remedial Action for an EQ
sp_question_ remedial_update	Update a suggested Remedial Action for an EQ
sp_question_update	Update an EQ
sp_question_update	Update an EQ

Table 3 below lists all stored procedures used in Assessment Administration: Evaluation Instrument (EI).

Table 3
Stored Procedures: Assessment Administration – Evaluation Instrument (EI)

Name	Description
sp_questionnaire_ available_question_grid	Data grid of Evaluation Questions not included in an EI
sp_questionnaire_ create	Create a new EI
sp_questionnaire_ question_create	Add a new Evaluation Question to the EI
sp_questionnaire_ question_update	Update an Evaluation Question in the EI
sp_questionnaire_ question_delete	Delete an Evaluation Question from the EI
sp_questionnaire_ question_grid	Data grid of Evaluation Questions included in an EI
sp_questionnaire_ question_read	Read the record of an Evaluation Question in the EI
sp_questionnaire_ question_score_update	Update the Evaluation Question Score in the EI
sp_questionnaire_ question_update	Update an Evaluation Question in the EI
sp_questionnaire_read	Read an EI record
sp_questionnaire_update	Update an EI

See a list of stored procedures for Assessment Delivery: Performance Evaluation (PE) in Table 4.

Table 4
Stored Procedures: Assessment Delivery Performance Evaluation (PE)

Name	Description
sp_evaluation_create	Create a new PE
sp_evaluation_delete	Delete a PE
sp_evaluation_read	Read a PE record
sp_evaluation_ score_grid	Datagrid of PE Scores
sp_evaluation_ score_read	Read a PEScore record
sp_evaluation_score_update	Update a PE Score
sp_evaluation_update	Update a PE

Table 5 contains a list of stored procedures for System Administration.

Table 5
Stored Procedures: System Administration

Name	Description
sp_login_email	Read Login Credentials for the e-mail
sp_login_lock	Lock a User Account
sp_login_validate	Validate Login Credentials
sp_user_create	Create a new User Account
sp_user_delete	Delete a User Account
sp_user_read	Read a User Account record
sp_user_update	Update a User Account

Data Views

Data grids in the Performance @ppraisal web application are filled by data views to join the necessary database tables. See a list of data views in Table 2 below.

Table 6
Data Views

Name	Description
v_area_list	Appraisal Areas
v_empirical_list	Empirical Data Sources
v_evaluation_list	Performance Evaluations
v_login	Authorized User Information
v_questionnaire_list	Assessment Instruments
v_question_list	Evaluation Questions
v_remedial_list	Remedial Actions
v_user_list	User Accounts

MODEL – VIEW – CONTROLLER DESIGN PATTERN

Performance @ppraisal was implemented as a three-tier web application by using the Model-View-Controller design pattern to separate the user interface (View) code from the business logic (Controller) and database operations (Model) See Figure 5 below.

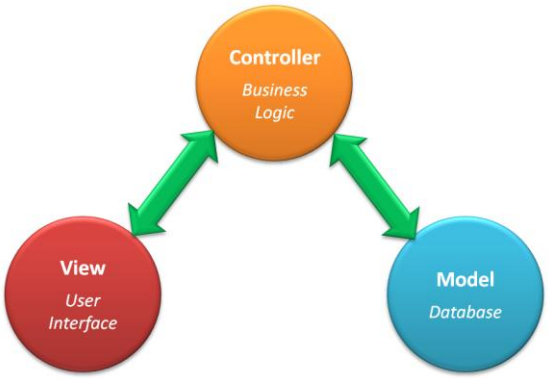

Figure 5
The Model-View-Controller Design Pattern

The directory structure to store the source code for the views and controllers is based on the design areas to make the source code easier to understand and maintain.

The controller implements the business logic for the Performance @ppraisal web application. One controller is defined for each use case specification. The controller will invoke the model functions and display the user-interface (views). Controllers are stored inside directories corresponding to the design areas. See Table 7 below for a list of controllers.

Table 7
Controllers

Name	Description
Area	Appraisal Area
question	Evaluation Question
questionnaire	Assessment Instrument
evaluation	Job Performance Evaluation
Review	Review Performance Evaluation
dev_plan	Development Plan
guide_score	Guidance – Evaluation Scores
guide_questionnaire	Guidance – Assessment Instrument

All database operations were implemented in models, one for each main database table. These models will update the data in the main database table and its related child tables. For instance, the question model will update the question database table and its related database tables: question_criteria and question_empirical. The question_criteria database table stores the minimal criteria to assign each one of the scores for a particular question. A question may be related to zero or more empirical data sources, and these relationships are stored in the related question_empirical database table. See Table 8 below for a description of each model.

Table 8
Models

Name	Description
Area	Appraisal Area
Email	E-mail Information
Evaluation	Job Performance Evaluation
Login	Login Credentials Validation
Question	Evaluation Question
Questionnaire	Assessment Instrument
stored_procedure	Execution of Stored Procedures
User	User Accounts

The user-interface is displayed in views that receive the data from the controller. See a snapshot of an Employee Job Performance Evaluation view in Figure 6 below.

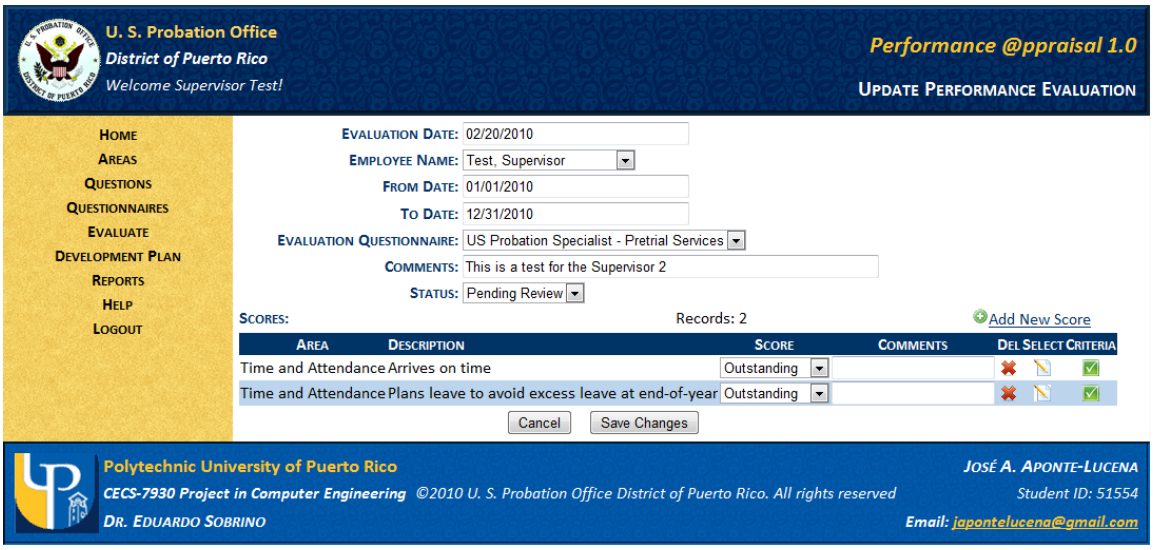

Figure 6

Employee Job Performance Evaluation View

See a list of data views used by Performance @ppraisal in Table 8 below.

Table 9

Views

Name	Description
area_create	Create a new Appraisal Area
area_list	List all Appraisal Areas
area_update	Update an Appraisal Area
empirical_create	Create a new Empirical Data Source related to an EQ
empirical_update	Update an Empirical Data Source related to an Evaluation Question
question_create	Create a new Evaluation Question
question_list	List all Evaluation Questions
question_update	Update an Evaluation Question
remedial_create	Create a new Remedial Action related to an Evaluation Question
remedial_update	Update a Remedial Action related to an Evaluation Question
question_create	Create a new Evaluation Question
question_criteria	Display the criteria to score an Evaluation Question
question_update	Update an Evaluation Question
questionnaire_create	Create a new Assessment Instrument
questionnaire_list	List all Assessment Instruments
questionnaire_update	Update an Assessment Instrument
evaluation_create	Create a new Performance Evaluation
evaluation_list	List all Performance Evaluations
evaluation_update	Update a Performance Evaluation
score_create	Create a new EQ Score
score_update	Update an Evaluation Question Score

All of these views use inherit from a global website template class to standardize the web application appearance. When this global template changes, the change is reflected on all pages of the Performance @ppraisal System application.

A directory structure based on the design areas helps to organize the large number of views needed by the Performance @ppraisal user interface. In addition, the performance evaluation e-mail notification uses an HTML body implemented as a view.

Software Tools

One of the most important goals for software development in the U. S. Probation Office is the ability to collaborate with agencies nationwide and to share with them applications developed locally. Since offices in different districts might have different programming platforms and environments, it is strongly suggested that new applications are developed using open-source programming tools that could be easily deployed across platforms. To maximize the software longevity, we installed the most recent version of the following software tools.

PHP 5.3.1

The business logic in the controllers was implemented in PHP version 5.3.1, an open-source scripting language to easily implement dynamic

HTML pages for web applications [3]. This version of PHP supports object-oriented programming.

Kohana 2.3.4

Kohana version 2.3.4 is a lightweight framework for PHP 5 that offers libraries and helpers to use the Model View Controller architectural pattern in PHP that supports object oriented programming [4]. It includes many useful helper libraries to simplify common programming tasks, such as the following: a SQL helper for database operations, an HTML helper to encapsulate the creations of tags, and a form helper to add fields to a form, among others.

In the software design phase, I evaluated different model-view-controller (MVC) frameworks before selecting Kohana as the best option. As a matter of fact, my first option was CodeIgniter, another MVC framework for PHP 4. It was easy to use, but did not work as expected with PHP 5.3. When I realized it will not work with the latest version of PHP, I discarded it, and migrated the code to Kohana. It represented a delay of two (2) weeks in software development. But I consider it a wise investment of time in the long run, considering the benefit of now having the latest versions of the software tools.

One drawback of the Kohana framework it does not work well with Microsoft Internet Information Server (IIS 5) due to the difficulty of implementing URL redirections in this web server. However, it works fine in an Apache web server.

MySQL Database Server 5.1

The Model (database) layer was implemented in a MySQL database server [5] version 5.1. Database operations were encapsulated in data views and stored procedures to optimize performance and execution time.

Javascript 1.8

The client-side programming of the views was implemented in JavaScript version 1.8, an object-oriented scripting language integrated as a component of the web browser. As a consequence, the actual version of Javascript depends on the version of the web browser running the application. Javascript greatly enhances the user interface of dynamic websites, for example, to implement the monthly calendar popup window to select dates in a form field.

PHP Documentor 1.4.3

PHP Documentor version 1.4.3 is a standard auto-documentation tool for the PHP language to create easy to read professional documentation from PHP source code in different pre-designed formats: HTML, PDF, Windows Helpfile CHM, and in Docbook XMLGuidance [7]. We generated a preliminary draft of the Performance @ppraisal's User's Guide with this tool by selecting the source code comments that we wanted to import, using special opening tag /** to surround the comment block.

The User's Guide was generated in HTML to be published as online help documentation with the Performance @ppraisal web application. See Figure 7 below for a sample page of the Performance @ppraisal User's Guide.

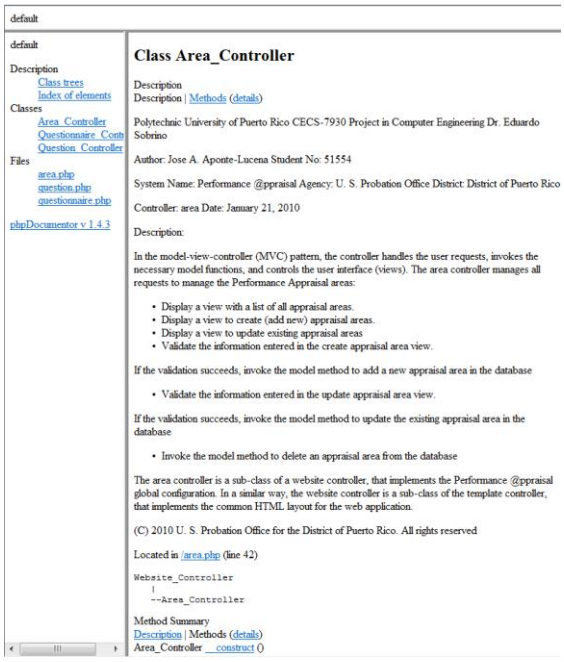

Figure 7
User's Guide Sample Page

jQuery 1.3.2

jQuery version 1.3.2 is a fast and concise JavaScript Library for rapid web development [6]. It simplifies event handling, animations, and Ajax interactions for rapid web development. In particular, we used it to implement the data grid appearance and behavior.

CSS 2

The color scheme and other style settings are implemented in a Cascading Style Sheet (CSS) Level 2 to facilitate the configuration of the look-and-feel of the Performance @ppraisal web application in a single file.

CONCLUSION

The Performance @ppraisal web application was implemented successfully as a three-tier architecture using the Model-View-Controller design pattern in PHP. As a result, the source code is easier to maintain when paired with good technical documentation and many comments in the source code.

In the positive side, using a robust software design, this application is scalable, easier to maintain and to incorporate new features in future releases. Software longevity was optimized by installing the latest versions of the selected software tools.

Collaboration was eased by selecting open-source tools to reduce costs and optimize compatibility across different platforms.

REFERENCES

[1] Administrative Office of the United States Courts, Office of Probation and Pretrial Services, "The Supervision of Federal Offenders", *Monograph 109*, March 2008, pp. 1-162.

[2] Administrative Office of the United States Courts, Office of Probation and Pretrial Services, *"Quality Performance Management"*, July 2006, pp. 1-9.

[3] The PHP Group, *"PHP"*, http://www.php.net.

[4] Kohana Team, *"Kohana: The Swift PHP Framework"*, http://www.kohanaphp.com.

[5] Oracle Corp., *"MySQL"*, http://www.mysql.com.

[6] Eichom, Joshua, *" PHP Documentor"*, www.phpdoc.org.

[7] The jQuery Project, *" jQuery: Write less, do more"*, http://www.jquery.com.