|  |
| --- |
| RIT Senior Project |
| CoVal2 |
| Design Document v0.4 |

|  |
| --- |
| Jonathon Leight, Trevin Maerten, Noel Nacion, John O’Connor2/23/2012 |

Table of Contents

[Introduction 3](#_Toc317767428)

[Purpose 3](#_Toc317767429)

[Background 3](#_Toc317767430)

[Stakeholders 3](#_Toc317767431)

[Quality Attributes 4](#_Toc317767432)

[Performance 4](#_Toc317767433)

[Availability 4](#_Toc317767434)

[Security 4](#_Toc317767435)

[Usability 5](#_Toc317767436)

[Portability 6](#_Toc317767437)

[Architectural Tactics 7](#_Toc317767438)

[Performance 7](#_Toc317767439)

[Availability 7](#_Toc317767440)

[Security 7](#_Toc317767441)

[Usability 7](#_Toc317767442)

[Portability 7](#_Toc317767443)

[Design Patterns 7](#_Toc317767444)

[Repository Pattern 8](#_Toc317767445)

[Views 9](#_Toc317767446)

[Database Design (Part 1) 9](#_Toc317767447)

[Element Catalog 9](#_Toc317767448)

[Context 10](#_Toc317767449)

[Architecture Background 10](#_Toc317767450)

[Database Design (Part 2) 11](#_Toc317767451)

[Element Catalog 11](#_Toc317767452)

[Context 12](#_Toc317767453)

[Architecture Background 12](#_Toc317767454)

[Database Design (Part 3) 13](#_Toc317767455)

[Element Catalog 14](#_Toc317767456)

[Context 14](#_Toc317767457)

[Architecture Background 15](#_Toc317767458)

[Model Class Diagram (Part 1) 16](#_Toc317767459)

[Element Catalog 16](#_Toc317767460)

[Context 17](#_Toc317767461)

[Architecture Background 17](#_Toc317767462)

[Model Class Diagram (Part 2) 18](#_Toc317767463)

[Element Catalog 18](#_Toc317767464)

[Context 19](#_Toc317767465)

[Architecture Background 19](#_Toc317767466)

[Model Class Diagram (Part 3) 20](#_Toc317767467)

[Element Catalog 21](#_Toc317767468)

[Context 21](#_Toc317767469)

[Architecture Background 21](#_Toc317767470)

[Controller and URL Mappings 22](#_Toc317767471)

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the application, and uses several different architectural views to portray varying aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made for the system.

This document may be used to communicate significant architectural decisions to the various stakeholders as needed. High level views of the architecture are described to indicate the overall structure of the project and the relationships among its components.

If architectural decisions change during development of the application, this documentation will provide a basis for revisiting those decisions.

Since the majority of this product’s functionality will be implemented in the database itself, the database design is described in higher detail than the actual application architecture.

## Background

Rochester Institute of Technology (RIT) currently uses a Co-op Evaluation System (CES) in order to keep track of what students did on their co-ops, as well as how the co-op program is enriching student’s educations. The current CES was developed almost 10 years ago. It was developed using technologies that had not been standard at the time, and used custom libraries that were not documented. Functionality was designed by two separate groups and then put together, leaving the application unorganized.

The new version of the CES, known here as CoVal2, will be a complete re-design and re-implementation of the CES. The implementation will use standard tools and standard libraries for easier maintenance. An emphasis will be placed on performance and usability for the design and implementation of CoVal2.

Due to the need for high performance, much of the work can, and will, be done on the application’s SQL server. This leads to an emphasis on the design of the database and its views and stored procedures rather than the design of the application.

## Stakeholders

The primary stakeholders for this application will be:

* Rochester Institute of Technology
* Office of Cooperative Education and Career Services (OCECS)
* RIT Students
* Co-op Employers
* RIT Faculty

# Quality Attributes

This section of the document explains what quality attributes were considered in the design of the CoVal2 system. It is a high level overview of the considerations and why they were considered. More detail about how they actually affected the design is covered in the next section.

## Performance

The CoVal2 system will be used mostly during the last three weeks of an academic term, as well as the next two weeks of the next academic term. The system will need to be able to support a large number of users during these times. In order for the system to be most useful, it needs to be able to quickly access and store the information that it deals with. As such, the main constraint that this architecture will focus on is that the system shall respond to requests within three seconds.

|  |  |
| --- | --- |
| **Source** | End User |
| **Stimulus** | Generate a request. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system must process the request and generate a response in no more than three seconds. |
| **Response Measure** | The response measure is the average time it takes to process a request on the system. |

## Availability

Since the CoVal2 system, as described in the Performance section of the document, will primarily be used for the three weeks at the beginning and end of a quarter, the system needs to be operational for most of that time. During the off-peak times in the middle of the quarter, availability is not quite as important. Overall, this leaves us with the requirement to maintain 95% uptime.

|  |  |
| --- | --- |
| **Source** | End User |
| **Stimulus** | Generate a request. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system must respond to the request. |
| **Response Measure** | The response measure is the percentage of the time that the system is available to handle the end user’s requests. |

## Security

Since the CoVal2 system will be dealing sensitive personal data, it must exhibit high security. The system must hide all database functionality from the user, and it must prevent unauthorized access. It must not, however, falsely prevent users from accessing the system. Any unauthorized access or intrusion attempt should be logged, if possible.

|  |  |
| --- | --- |
| **Source** | Unauthenticated End User |
| **Stimulus** | Attempting to view secure data. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will redirect the user to a login page, and then, upon authentication, return them to the page they were trying to view. |
| **Response Measure** | The response measure is the number of pages that are incorrectly unsecured. |

|  |  |
| --- | --- |
| **Source** | Authenticated End User |
| **Stimulus** | Attempting to view secure data they are authorized to view. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will display the data to the user. |
| **Response Measure** | The response measure is the number of pages that do not display data to an authorized user when it should. |

|  |  |
| --- | --- |
| **Source** | Authenticated End User |
| **Stimulus** | Attempting to view secure data they are not authorized to view. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will display a message indicating that they are unauthorized to view the requested data. |
| **Response Measure** | The response measure will be the number of pages of data that are displayed to unauthorized users. |

## Usability

Due to the relative infrequency of use of the system, the interface needs to be easily usable and understandable in order for people to learn to use it quickly, get their work done with the system, and then not use the system again for at least another academic term. Some users of the system will use the same features many times during their approximately three weeks of usage, so while the usability of those features is important, it is less important, so long as it does not affect their productivity.

|  |  |
| --- | --- |
| **Source** | End User |
| **Stimulus** | Attempt to access key areas of the system. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will allow access to key areas in less than five mouse clicks. |
| **Response Measure** | The response measure will be the average number of clicks required to access parts of the system. |

|  |  |
| --- | --- |
| **Source** | End User |
| **Stimulus** | Attempt to navigate the system. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will allow for navigation of the user interface in an intuitive, web-standard way. |
| **Response Measure** | The response measure will be the number of people who cannot find what they are looking for in the system within 60 seconds. |

## Portability

Due to the large user base of the system, a large number of end user systems must be supported. This will require the user interface be designed and implemented in a standards compliant way so that users can get the most out of it.

Due to a smaller user base, and a smaller system base, the evaluator and administrator portions of the system are slightly less strict on what needs to be supported. Administrators and evaluators will be able to use any standards compliant web browser to access the system. These include, but are not limited to: Chrome, Firefox 3.6+, Internet Explorer 8+, Safari 5+, and Opera.

The large user base and large system base for employers and students will require that the system function on all of the above web browsers, as well as Internet Explorer 6+.

|  |  |
| --- | --- |
| **Source** | End User |
| **Stimulus** | Attempt to use the system in an unsupported browser. |
| **Artifact** | System |
| **Environment** | Runtime |
| **Response** | The system will display a warning that the user is using an unsupported browser, and that some functionality may not work correctly. |
| **Response Measure** | The response measure will be the number of actions that are attempted in a web browser that fail to work due to browser incompatibilities. |

# Architectural Tactics

This section of the document explains the actual impact that the quality attributes had on the design and architecture of the application.

## Performance

By utilizing the third normal form and removing any redundant data, there is less data to parse through when querying the database for information. The stored procedures used by CoVal2 aid in data retrieval and decrease the bandwidth used during querying by limiting the amount of communication between the client and the server. Using stored procedures and the database to their full potential will help increase the performance of the system.

## Availability

The system is designed to continue running with minimal maintenance during most of the time. The system only requires high availability during the first few weeks of a new period, and the system will provide ample availability during this time. A focus on error detection and prevention will help to make sure that the application is always available when it needs to be.

## Security

All access is granted based on a user’s type. An unregistered user cannot access the system or its data in any way. User passwords are stored only as a hash of the password in the database to ensure no one besides the user knows their password. Session information is stored as a cookie on the user’s machines and cannot be accessed by anyone but that user during that session. Particular attention will be paid to user roles in the system, and all controllers and actions will be secured in order to prevent unauthorized access.

## Usability

CoVal2 has been designed with a number of usability features, specifically in the user interface, but there are a number of features that also need to be accounted for in the backend of the system. CoVal2 will include an auto-saving feature that will save user’s input automatically when filling out forms. This is taken into consideration in actions that can act on either the full set of data, or just a partial set of data.

## Portability

CoVal2 is designed to run on all major, up-to-date browsers including but not limited to Mozilla Firefox, Microsoft Internet Explorer, and Google Chrome. This means that the user interface will be written in standards compliant HTML and CSS, using widely supported JavaScript libraries, and very little custom JavaScript that must be maintained by the OCECS.

## Design Patterns

This is not a comprehensive list of design patterns used in the system. It does, however, include the patterns that we think are most important for helping with our architectural tactics.

### Repository Pattern

The repository pattern is a common pattern in the C# and MVC3 community. It allows for a decoupling of the model objects and the business logic layer of the application. This allows for more flexibility in the way that data is stored in the system.

Repository implementations can be swapped out, so objects can be stored in any type of storage, and a change to the storage engine will only require a repository to be swapped out.

The repository pattern also allows for dependency injection to be used, and helps make unit testing easier since unit tests do not have to rely on the database.

Each of our model objects will be populated through the use of a Repository.

# Views

## Database Design (Part 1)



Figure : Session, Configuration, and Employer Password tables.

### Element Catalog

Other than the session tables, these tables are not really connected in any way. The separation of these tables was just to make the design easier to understand.

1. EmployerPassword
Contains all information to save an employer’s password.
2. Configuration
Contains a configuration value and the time the configuration was last updated.
3. Session
Contains information about a session, including the user’s IP address, the time they started using the system, the time their session was last updated, and their user agent. Two types of sessions must be maintained: DCE sessions and Employer sessions. These are identified by corresponding entries in the DCESession and EmployerSession tables.
4. DCESession
Links the DCE of an RIT user to their session.
5. EmployerSession
Links the employer’s ID to their session.

### Context

This database diagram is part of the CoVal2 database. Since MSSQL does not allow for cross-database foreign keys, there is no foreign keys to the OCECS database, but there is, in spirit, a link between employerId, contactId, and the Contacts.id field in the OCECS database.

### Architecture Background

The EmployerPassword table was designed with usability in mind. It allows for an employer to have their own custom password, and one single account, rather than having multiple accounts and passwords for each evaluation that they need to fill out.

The Session tables were designed with usability in mind as they allow students and employers to log in to the system and maintain their session rather than being kicked out of the system as soon as they close their browser or their CoVal2 tab.

## Database Design (Part 2)



Figure : Event Log and Email Notification tables

### Element Catalog

The tables in this part of the database design all revolve around the event log and the email notifications that will be sent out.

1. LogEvent
A log entry representing an event in the system including a date and a time, a message explaining the event, which user caused the event, the type, and the severity of the event.
2. LogEventType
This table contains a list of all of the different event types that can occur in the system.
3. LogEventSeverity
This table contains a list of all of the different severity levels a log entry can be.
4. EmailLogEvent
Information about an email, including the subject line, sender, receivers, and a link to the content.
5. EmailContent
The content of an email, including the subject line, header, footer, and the time it was last updated.
6. EmailType
The type of an email.
7. DepartmentEmailContent
Special department specific information in an email.

### Context

The tables in this diagram are all part of the CoVal2 database. The tables, except for LogEvent, are all self-contained and do not have foreign keys to any other system. The userId field of the LogEvent table will link to the actual user that caused the event.

### Architecture Background

Given that the system must be both performant and available, it was decided that there should be an event log in the system. This allows for an easier time figuring out if there are errors with the system or if something is going on to make the system less performant.

Given that the system is supposed to be usable, the email related tables have been designed so that departments can change their own email notifications, and that changes are kept separate from the overall body of the email messages. This allows for modified email messages to be easily detected, and allows for modification of the overall message without breaking department specific modifications and vice versa.

## Database Design (Part 3)



Figure : Form related tables

### Element Catalog

The tables in this section of the database deal with the Form, Questions, and Evaluations.

1. Evaluation
Contains references to the placement, student work report, employer evaluation, evaluation status, student status, and the times the student and employer were last updated.
2. EvaluationStatus
The status of an evaluation. This is linked twice as there are statuses for both the student and employer evaluation.
3. EvaluationApprovals
Contains a link to an evaluation and it’s status. Contains who approved the evaluation.
4. EvaluationApprovalStatus
Contains the status of an evaluation and a link to an evaluation.
5. Form
Contains the name of the form.
6. DepartmentQuarterForm
Specifies which form should be used for which department in which quarter.
7. FormQuestions
Contains links to the form, the surrounding questions, the question itself, and the question category.
8. Question
The question and the question’s type.
9. QuestionType
The type of a question. This includes, but is not limited to: text, number, date, Lykart, and double Lykart.
10. QuestionAnswer
Contains a serialized answer to a question and the time it was last updated. Also contains links to the question, the evaluation, and the FormQuestions join table.

### Context

The tables in this diagram are all part of the CoVal2 database. The only links to external tables are the approvedBy column in EvaluationApprovals, and the placemendId in the Evaluation table.

### Architecture Background

These tables were designed mostly with performance in mind. They were designed to third normal form to try and reduce the duplication of data, and indices were added where they would increase performance during queries.

Binary serialization for the answers was chosen due to the large number of different possible answer types that could exist in the system. Rather than making a large number of tables that linked together in complex ways, a single table with a binary column was chosen to represent the answers. Given that the system needs to be performant, CLR stored procedures have been considered to allow the database to process the binary data without having to query all of the data, send it to an application, deserialize everything, and then finally be able to work with the data.

Since the system still needs to be usable, CLR stored procedures will be made that allow the database to be queried directly without the use of an application capable of deserializing the binary data. This will allow for the most flexibility, with little to no performance cost.

## Model Class Diagram (Part 1)



Figure : Session, Configuration, and Employer information

### Element Catalog

1. Employer
Contains all information related to an employer.
2. EmployerPassword
Each employer may have their own password, so this structure is used to modify the password entry in the database.
3. Configuration
Contains a configuration value and the time the configuration was last updated.
4. Session
Contains information about a session, including the user’s IP address, the time they started using the system, the time their session was last updated, and their user agent.
5. DCESession
Subclass of Session that includes the DCE username of the session’s user.
6. EmployerSession
Subclass of Session that includes the id of the employer.

### Context

The model classes displayed in this diagram are in the CoVal.Models namespace.

### Architecture Background

Using subclassing on the Session class allows for easy distinguishing of whether the session belongs to an employer or an RIT user. This allows for increased performance, as well as less data duplication.

## Model Class Diagram (Part 2)



Figure : Event Log and Email Notification information

### Element Catalog

1. LogEvent
The log of an event.
2. LogEventType
The type of an event.
3. LogEventSeverity
The name of an event’s severity.
4. EmailLogEvent
Information about an email, including the subject line, sender, receivers, and a link to the content.
5. EmailContent
The content of an email, including the subject line, header, footer, and the time it was last updated.
6. EmailType
The type of an email.

### Context

The model classes displayed in this diagram are in the CoVal.Models namespace.

### Architecture Background

Since the system must be performant, enum data types were chosen for the “type” classes since each type will end up being only a single instance on the server, and comparison can be made against the types rather than using, for example, string comparisons, which are slower and more processor intensive.

## Model Class Diagram (Part 3)



Figure : Form related information

### Element Catalog

1. Evaluation
Contains references to the placement, student work report, employer evaluation, evaluation status, student status, and the times the student and employer were last updated.
2. EvaluationStatus
The status of an evaluation.
3. EvaluationApprovals
Contains a link to an evaluation and it’s status. Contains who approved the evaluation.
4. EvaluationApprovalStatus
Contains the status of an evaluation and a link to an evaluation.
5. Form
Contains the name of the form.
6. DepartmentQuarterForm
Links a quarter with a specific form for a department.
7. FormQuestions
Contains links to the form, the surrounding questions, the question itself, and the question category.
8. Question
The question and the question’s type.
9. QuestionType
The type of a question.
10. QuestionAnswer
Contains a serialized answer to a question and the time it was last updated. Also contains links to the question, form question, and evaluation.

### Context

The model classes displayed in this diagram are in the CoVal.Models namespace.

### Architecture Background

Enum types were once again chosen for all “type” related classes for performance reasons. Creating subtypes of QuestionAnswer also allows for easy and fast determination of whether an answer was filled in by a student or an employer.

## Controller and URL Mappings

This section of the document shows what controllers will be used in the system, as well as what actions they perform, and what each action will be doing. Actions that do not specify GET or POST shall be assumed to be GET.

|  |  |
| --- | --- |
| **URL** | **Description** |
| / | Displays the homepage. |
| /Home | Same as /. |
| /Login | Displays a login page on GET, attempts to authenticate the user on POST. |
| /Logout | Logs the user out. |
| /Form | Displays a list of all of the forms in the system with links to the CRUD functions. |
| /Form/Details/# | Shows what the form would look like if it was being filled out. |
| /Form/Create | Displays a page allowing for the creation of a new form on GET, creates the form on POST. |
| /Form/Edit/# | Displays a page where the specified form can be edited on GET, edits the form on POST. |
| /Form/Delete/# | Displays a page confirming the form deletion on GET, deletes the form on POST. |
| /Employer | Lists information about the currently logged in employer. Links to the lost password feature. |
| /Employer/Edit | Displays a page allowing the current employer to change his password on GET, changes the password on POST. |
| /Employer/LostPassword | Displays a page allowing an employer to specify their email address so that they can set a new password if they forget their current one on GET, sends out the email with a token on POST. |
| /Employer/Confirm | Displays a page where the employer can specify a token allowing them to set their initial password or set a new one via the lost password feature on GET, sets the password on POST. |
| /Question | Displays a search page allowing the user to search for a question in the system. |
| /Question/Search | Displays the list of search results. |
| /Question/Details/# | Displays what the question would look like if it was being displayed on a form. |
| /Question/Create | Displays a form allowing the user to create a new question on GET, creates the new question on POST. |
| /Question/Edit/# | Displays a page allowing the user to edit the question on GET, edits the information on POST. |
| /Question/Delete/# | Displays a confirmation page or an error page on GET, deletes the question if possible on POST. |
| /Evaluation | Displays a search page that allows a user to search for an evaluation entry in the system. |
| /Evaluation/Search | Displays the list of search results. |
| /Evaluation/Details/# | Displays all of the details of the specified evaluation. |
| /Evaluation/Edit/# | Allows for the changing of statuses for the evaluation on POST. |
| /Submission | Displays the equivalent of the student and employer homepage in the current CES system. |
| /Submission/Details/# | Displays the submission without editable responses. |
| /Submission/Edit/# | Displays the form in an editable format for the student or employer to fill out the form on GET, saves answers or submits the form on POST. |
| /Notification | Displays a list of notifications for administrators, or the notifications for a specific department for evaluators. |
| /Notification/Details/# | Displays a specific notification as it would appear if it was sent via email. |
| /Notification/Edit/# | Displays a page allowing the user to edit either the whole notification, or just the department specific information on GET, edits the notification on POST. |
| /Notification/Send | Displays a list of possible notifications that can be sent on GET, and allows for a notification to be sent on POST. |
| /Report | Displays a search form allowing the user to search for evaluations to report on. |
| /Report/Search | Displays a list of questions that can be reported on after receiving the search parameters. |
| /Report/Details/#params | Displays the results of a report based on the specified parameters. Based on the parameters, may return the result as HTML or Excel format. |
| /Log | Displays an event log to the user. Also shows a search form. |
| /Log/Search | Displays the search results. |
| /Log/Details/# | Displays the details of a specific event to the user. |