

|  |  |  |
| --- | --- | --- |
| Document name | Architecture Document  Novartis\_iPortal | System Recruitment Specification |
| Version no. | 0.4 | 1.2 |
| Release date | 18-Dec-13 | 1st Oct ‘09 |
| Document ID | iPortal/AD |  |
| Classification | Confidential |  |

This document of Cybage Software Pvt. Ltd. is for restricted circulation. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means – recording, photocopying, electronic and mechanical, without prior written permission of Cybage Software Pvt. Ltd.

|  |
| --- |
| Architecture Document (AD) |

**Document History**

| Ver. No | Release Date | Created By / Modified By and Date | Reviewed By and Date | Approved By and Date | Remarks and Changes Made |
| --- | --- | --- | --- | --- | --- |
| 0.1 | 27-Nov-13 | Trupti Jangam / 27-Nov-13 |  |  | First draft |
| 0.2 | 04-Dec-13 | Trupti Jangam / 04-Dec-13 |  |  | Second draft |
| 0.3 | 11-Dec-13 | Trupti Jangam / 11-Dec-13 |  |  | Third draft |
| 0.4 | 18-Dec-13 | Trupti Jangam / 18-Dec-13 |  |  | Fourth draft |

**Distribution List**

| Sr. No. | Name | Role | Organization Name |
| --- | --- | --- | --- |
|  | Abhijay Datta | Associate Director | Novartis |
|  | Eric Chiu | Solution Architect | Novartis |
|  | Wilson Furtado | Delivery Manager | Cybage |
|  | Ashish Rathi | Project Manager | Cybage |

Table of Contents

[1. Introduction 4](#_Toc366233181)

[1.1 Purpose of the Document 4](#_Toc366233182)

[1.2 Objective of Architecture Document 4](#_Toc366233183)

[1.3 Scope of Architecture Document 4](#_Toc366233184)

[1.4 Acronyms and Definitions 4](#_Toc366233185)

[1.5 References 5](#_Toc366233186)

[2. Architecture Goals and Constraints 5](#_Toc366233187)

[3. Physical Architecture 5](#_Toc366233188)

[3.1 Physical Architecture 5](#_Toc366233189)

[3.1.1 Hardware Requirements 5](#_Toc366233190)

[3.1.2 Software Requirements 5](#_Toc366233191)

[3.1.3 Network Requirements 5](#_Toc366233192)

[3.1.4 Data Center Requirements 5](#_Toc366233193)

[4. Logical Architecture 5](#_Toc366233194)

4.1 [Pattern Used - MVC 5](#_Toc366233195)

[4.2 Architecture Diagram 8](#_Toc366233195)

[4.3 Logical Sub-system 8](#_Toc366233196)

[4.3.1 Activation & Login 8](#_Toc366233197)

[4.3.2 Document Management 12](#_Toc366233197)

[4.3.3 SafetyNotification Management 18](#_Toc366233197)

[4.3.4 Adaptive Monitoring Management 27](#_Toc366233197)

[4.4 Deployment and Build Script 28](#_Toc366233196)

[5. Process View 29](#_Toc366233198)

[6. Architectural Strategies 29](#_Toc366233199)

[6.1 Tools/RoR Frameworks Used 29](#_Toc366233200)

[6.2 Technoloy Stack & Third Party Gems 30](#_Toc366233200)

[6.3 Session Handling 38](#_Toc366233200)

[6.4 Exception Handling 39](#_Toc366233200)

[7. Appendix 39](#_Toc366233199)

# Introduction

This document will be used for depicting all the Architectural designs of **iPortal** project developed for client **Novartis**.

## Purpose of the Document

The purpose of this document is to

* Identify various components/objects in the sub-system to granular level
* Identify core modules/sub-systems of the system and sub-system boundary.
* Design detailed class diagram, sequence diagram, activity diagram, and so on.
* Identify database design (physical, logical, and more).
* Identify data structures and algorithm.
* Identify various R&D issues.
* Identify the technology stack required to deploy the application, and document the build/package/deployment process

## Objective of Architecture Document

The objectives of AD are to capture and convey all the important architectural decisions taken.

## Scope of Architecture Document

The scope of AD is <Mention the area of the system, which is covered in this AD>

## Acronyms and Definitions

This sub-section includes the definitions of all acronyms required to interpret the AD properly.

| Sr. No. | Acronyms | Definitions |
| --- | --- | --- |
|  | Imp | Investigational Medicinal Product |
|  | CPO | Country Pharmaceutical Organization |
|  | SN/Sn | Safety Notification |
|  | Sn\_version | Safety Notification Version |
|  | Sn\_type | Safety Notification Type |
|  | Sn\_country\_ack | Safety Notification Acknowledgement |
|  | Sn\_site\_ack | Safety Notification Site Acknowledgement |
|  | Sn\_study\_country\_ack | Safety Notification Study Country Acknowledgement |
|  | Ccra | Central CRA |
|  | Fm | Field Monitor |

## References

This sub-section provides a complete list of all documents referenced elsewhere in the AD with the details of the sources from which the references can be obtained.

# Architecture Goals and Constraints

<Not fill yet>

# Physical Architecture

This section describes the physical architecture of the system along with hardware and software requirements. The IS team and Hosting\Deployment team are the major stake holders of this section.

## Physical Architecture

### Software Requirements

<Not fill yet>

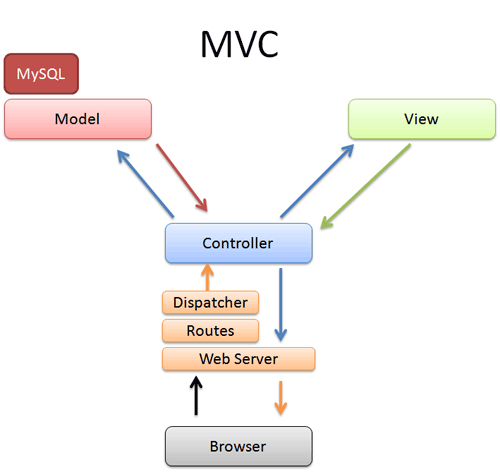
### Network Requirements

<Not fill yet>

# Logical Architecture

## Pattern used – MVC

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. Following diagram describes MVC pattern in **Ruby on Rails** application.



### Model (ActiveRecord )

Maintains the relationship between Object and Database and handles validation, association, transactions, and more.

This subsystem is implemented in ActiveRecord library which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. Ruby method names are automatically generated from the field names of database tables, and so on. ActiveRecord is a GEM and it is located under rvm dependencies. Whenever rails is installed, ActiveRecord gets automatically installed as its runtime dependency.

### View ( ActionView )

A presentation of data in a particular format is triggered by a controller's decision to present the data. They are script based templating systems like JSP, ASP, PHP and very easy to integrate with AJAX technology.

This subsystem is implemented in ActionView library which is an Embedded Ruby (ERb) based system for defining presentation templates for data presentation. Every Web connection to a Rails application results in the displaying of a view. ActionView is a GEM and it is located under rvm dependencies. It is a component of ActionPack GEM. Whenever rails is installed, ActionPack gets automatically installed as its runtime dependency.

### Controller ( ActionController )

The facility within the application directs traffic, on the one hand querying the models for specific data, and on the other hand organizing that data (searching, sorting, and messaging it) into a form that fits the needs of a given view.

This subsystem is implemented in ActionController which is a data broker sitting between ActiveRecord (the database interface) and ActionView (the presentation engine).

ActionController is a GEM and it is located under rvm dependencies. It is a component of ActionPack GEM. Whenever rails is installed, ActionPack gets automatically installed as its runtime dependency.

When user requests the url, controller parses the request, route and dispatch it to the appropriate method with the help of dispatcher, asks for model data and gives that data to view component. Controller also manages sessions, caching of data, view helpers.

*Directory Representation for Rails MVC Framework:*

Assuming a standard, default installation over Linux, ActiveRecord, ActionView and ActionController libraries/gems can be find like this:

$> cd /usr/local/lib/ruby/gems/1.9/gems

$> ls

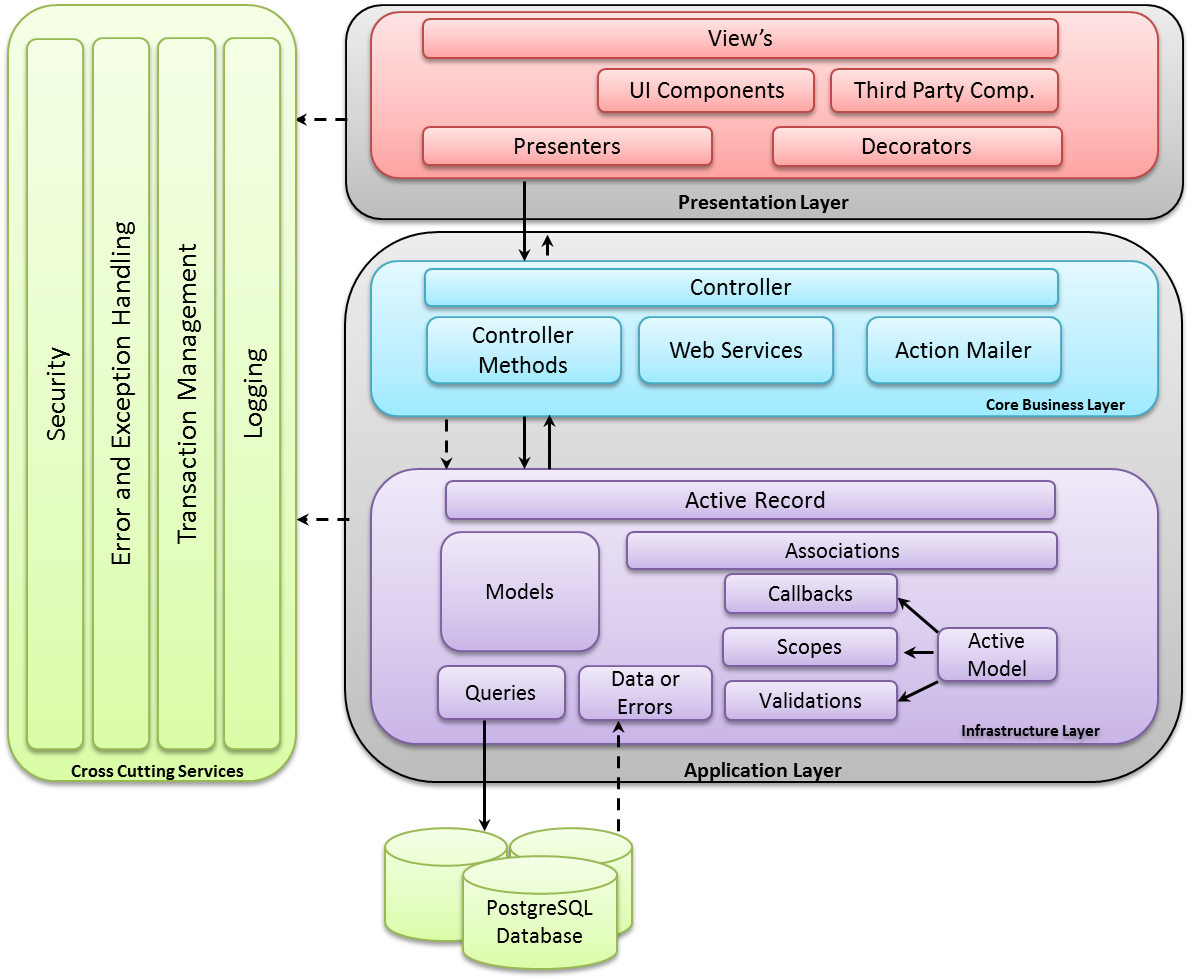
Many sub-directories including following will be seen:

* actionpack-x.y.z
* activerecord-x.y.z
* rails-x.y.z

activerecord-x.y.z is the actual ActiveRecord library.

While ActionView and ActionController can be find under actionpack-x.y.z library.

## Architecture Diagram



## Logical Sub-system

A Logical System may have one or more sub-systems associated with it. Following section will list all the identified subsystem and describe each individual subsystem.

### Account Activation & Login

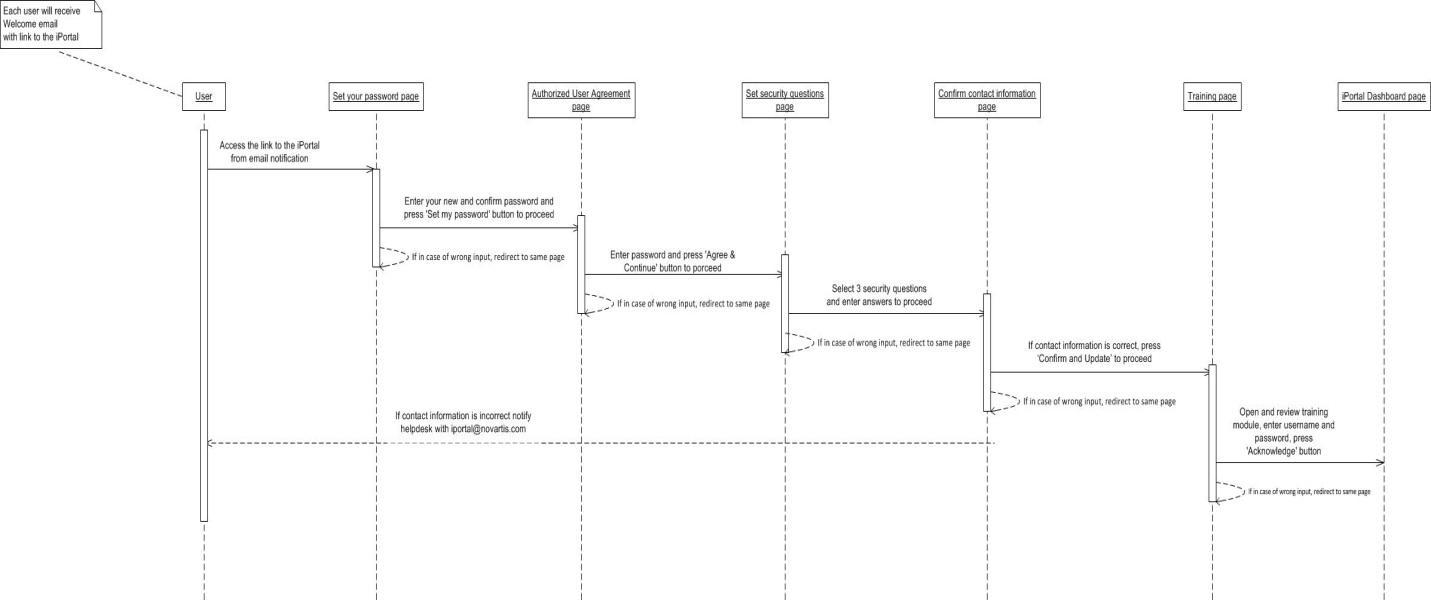
Account Activation Flow Diagram



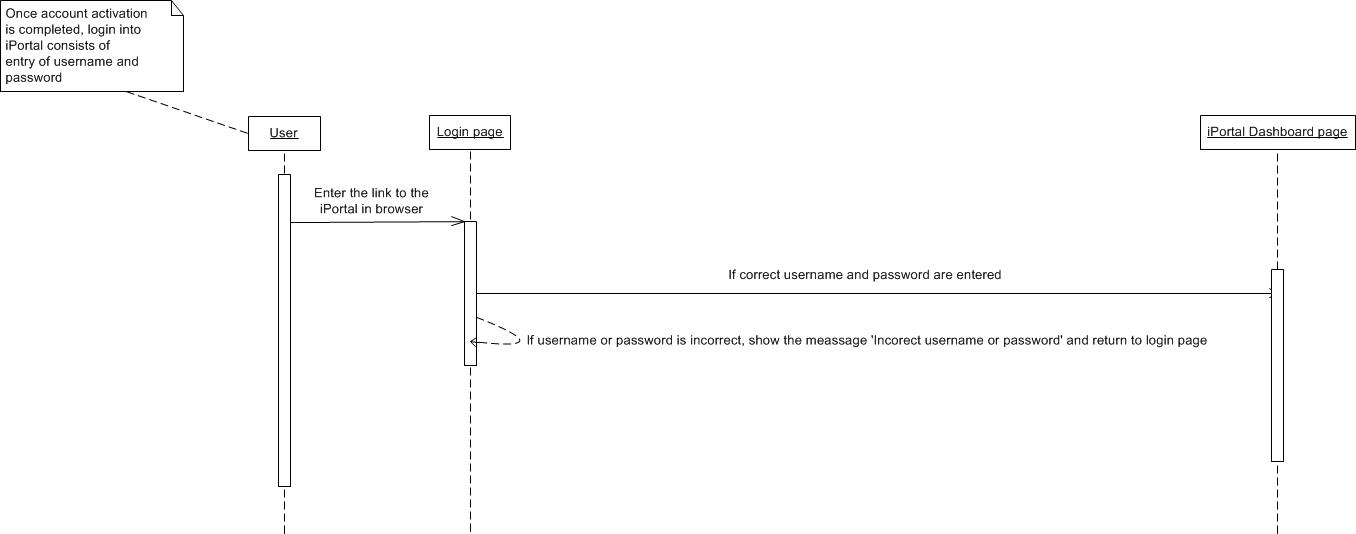
Login Flow Diagram



Account Activation Sequence Diagram



Login Sequence Diagram



Component Description for Account Activation

<Not fill yet>

Component Description for Login

Please refer following document for more description about classes, methods and models involved in Login.



### Document Management

#### Database and Data Requirements

##### Database ER Diagram



#### Object Model & Interface

*Class Diagram*



*Class Description*

*Document Class Description*

For Document class description refer below document.



#### Flow and Sequence Diagram

Flow Diagram



Sequence Diagrams

For All Users



For Study



For Imp



For Study Country



For Site



For Role



#### Component Description

Please refer following document for more description about classes, methods and models involved in Document Upload.



### SafetyNotification Management

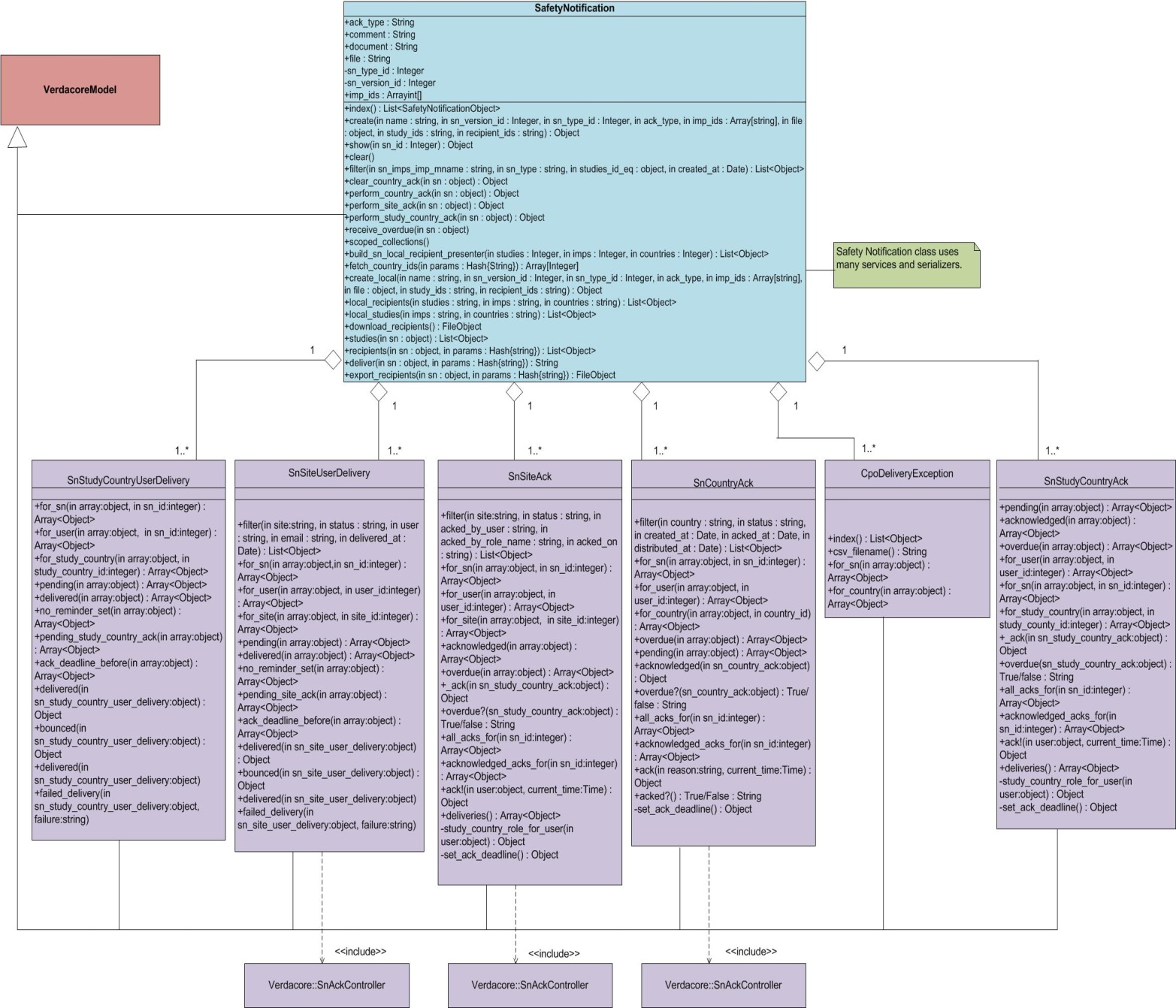
#### Database and Data Requirements

##### Database ER Diagram



#### Object Model & Interface

*Class Diagram*



*Class Description*

For Safety Notification class description refer below document.

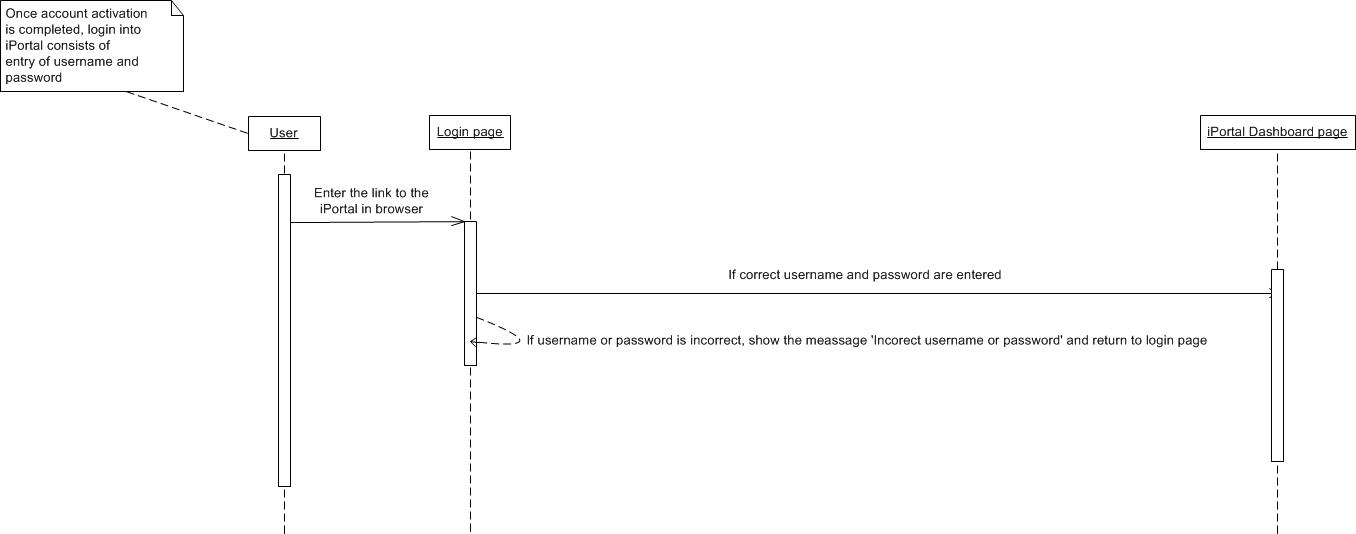
**

*Sequence Diagrams for Safety Notification Management*

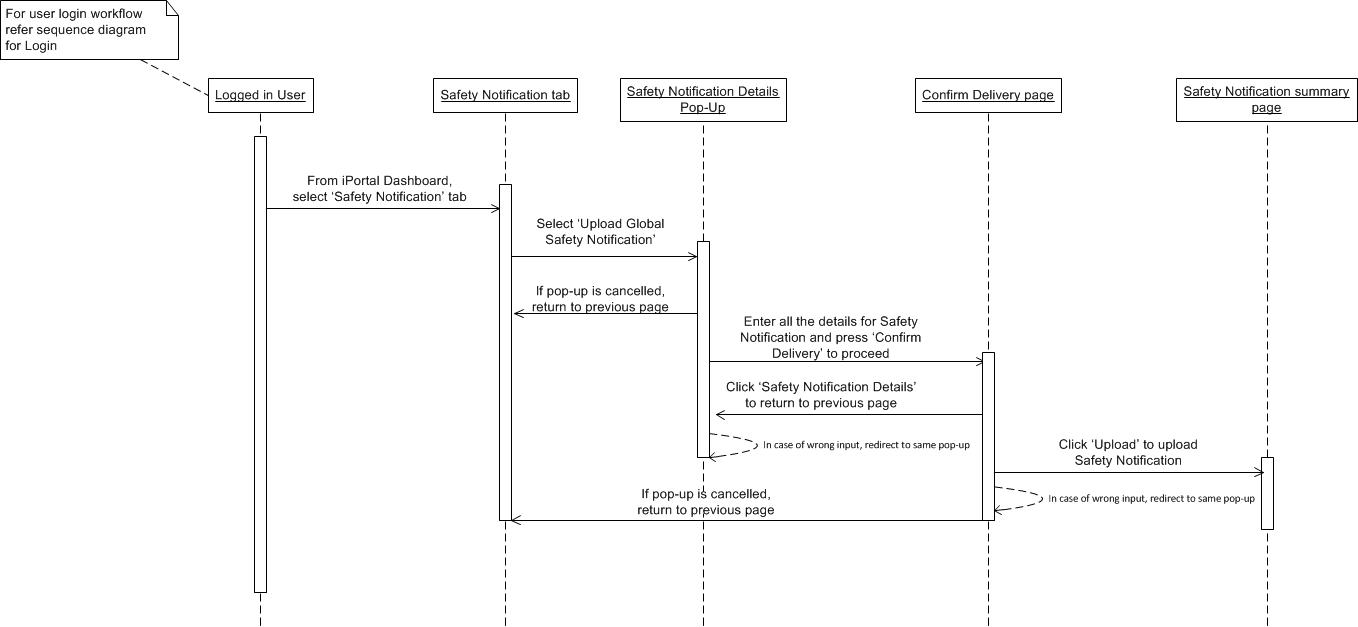
Account Activation Sequence Diagram :



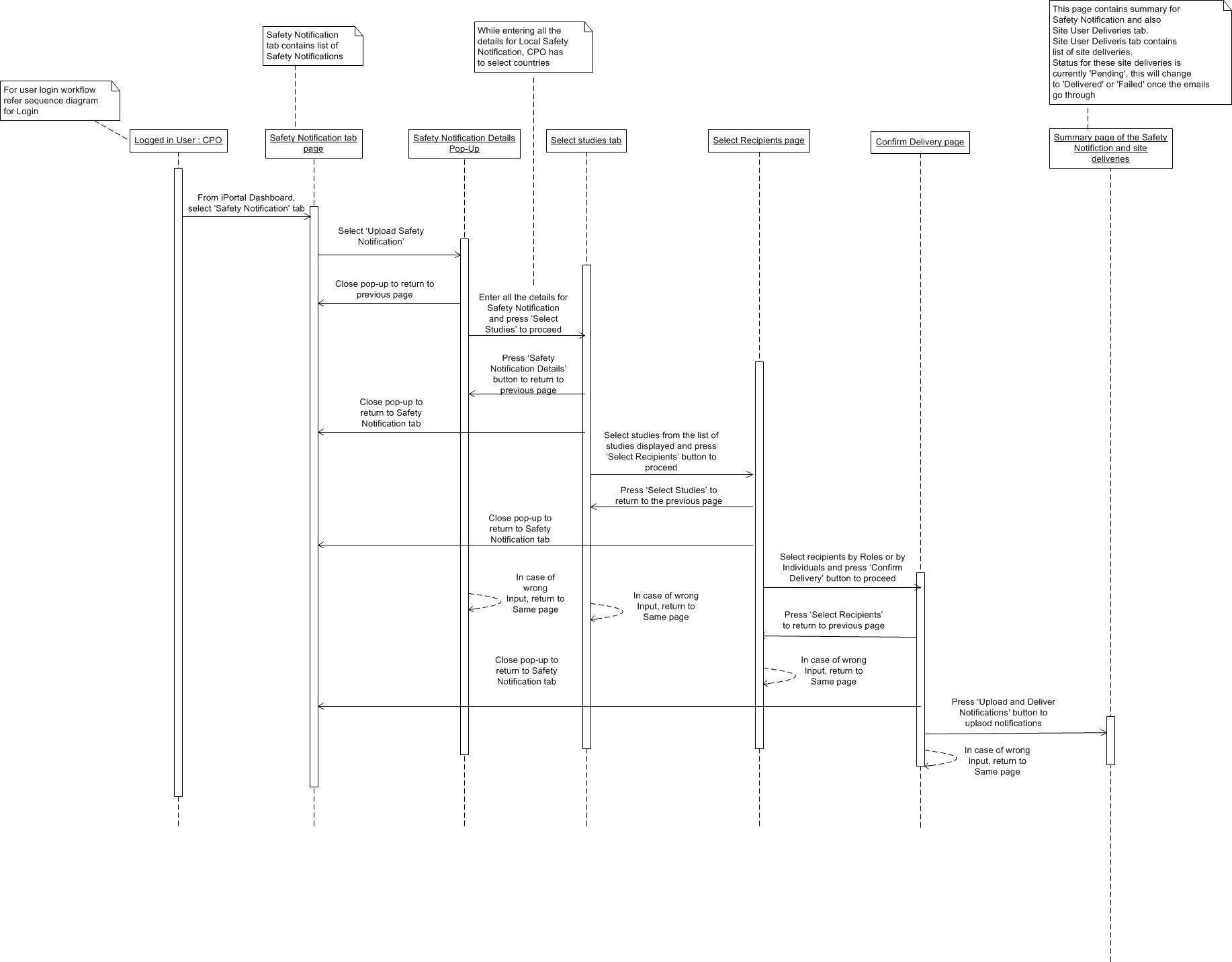
Login Sequence Diagram :



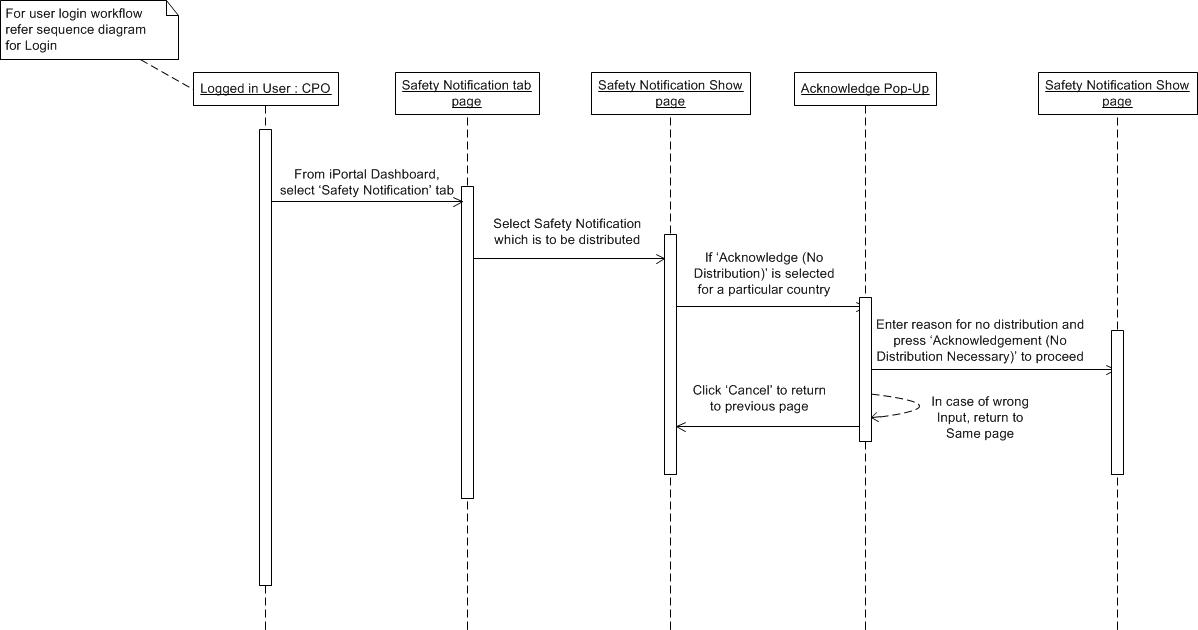
Global Safety Notification Upload Sequence Diagram :



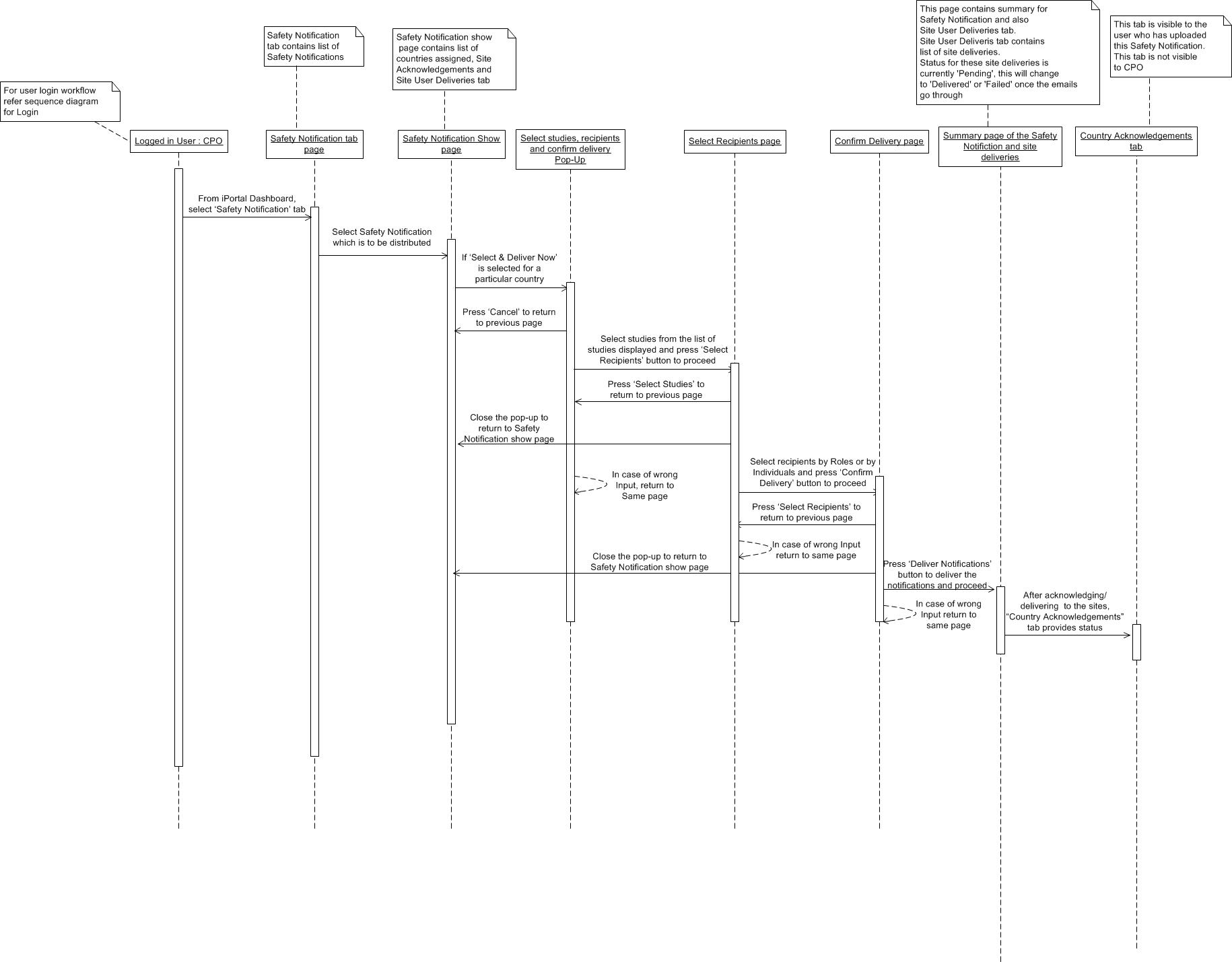
Local Safety Notification Upload Sequence Diagram :



CPO Acknowledgement for No Distribution Sequence Diagram :



CPO Acknowledgement for Select & Distribute Sequence Diagram :



### Adaptive Monitoring Management

#### Database and Data Requirements

##### Database ER Diagram



#### Object Model & Interface

*Class Diargam*



## Deployment and Build Script

For Deployment script refer Deployment Script document.

For Environment setup & Build script refer Environment & Build Script document.



# Process View

# Architectural Strategies

This section describes features or strategies used in designing the architecture of the application. These strategies also provide insight into the key abstractions and mechanisms used in the system architecture. Following are few strategies which can be mentioned in this section:

## Tools/RoR Frameworks Used

This section will describe any tools/RoR frameworks to be used in the application.

| Framework | Environment ( Runtime/Testing) | Description |
| --- | --- | --- |
| ActiveAdmin | Runtime | ActiveAdmin is the administration framework for business critical Ruby on Rails applications. It is used for generating administration style interfaces. It abstracts common business application patterns to make it simple for developers implement beautiful and elegant interfaces with very little effort. Allow developers to customize every nook and cranny of the interface easily. |
| Jquery | Runtime | jQuery is a fast, small, and feature-rich JavaScript framework. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. |
| Capistrano | Runtime | Capistrano is a remote server automation tool. It supports the scripting and execution of arbitrary tasks, and includes a set of sane-default deployment workflows. It reliably deploy web application to any number of machines simultaneously, in sequence or as a rolling set. |
| Capybara | Testing | Capybara is acceptance test framework for rack based web applications. Capybara helps you test web applications by simulating how a real user would interact with your app. It is agnostic about the driver running your tests and comes with Rack::Test and Selenium support built in. WebKit is also supported through an external gem. |
| Cucumber | Testing | Cucumber is Behavior Driven Development framework. Cucumber lets software development teams describe how software should behave in plain text. The text is written in a business readable domain specific language and serves as documentation, automated tests and development-id, all rolled into one format. |
| Shulda | Testing | Shulda make tests easy shoulda-context and shulda-matchers. Shoulda's contexts make it easy to write understandable and maintainable tests for Test::Unit. It's fully compatible with your existing tests in Test::Unit, and requires no retooling to use.  shoulda-matchers provides Test::Unit- and RSpec-compatible one-liners that test common Rails functionality. These tests would otherwise be much longer, more complex, and error-prone. |

## Technology Stack Used & Third Party Gems

Technology Stack:

A technology stack comprises the layers of components or services that are used to provide a software solution or application. iPortal application uses following technology stack.

* Language – Ruby
* Web Framework – Rails
* Web Server – Thin, Unicorn
* Relational Database – Postgresql
* Operating System – Linux
* Cloud Pass Solution - EngineYard

Third party Gems

The below section describes third party gems/libraries used in the system.

| Gem Name | Version | Environment (Development, Test, Production) | Description |
| --- | --- | --- | --- |
| rails | 3.2.14 | All | Rails are a web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. |
| pg | - | All | Pg is the Ruby interface to the PostgreSQL RDBMS. It works with PostgreSQL 8.4 and later. |
| active\_admin | - | All | Active Admin is a Ruby on Rails framework for creating elegant backends for website administration. |
| devise | 2.2.4 | All | Devise is a flexible authentication solution for Rails |
| nokogiri | 1.5.10 | All | Nokogiri is an HTML, XML, SAX and Reader parser. |
| active\_model\_serializers | 0.8.0 | All | ActiveModel::Serializer implementation and Rails hooks |
| carrierwave | - | All | This gem provides a simple and extremely flexible way to upload files from Ruby applications. It works well with Rack based web applications, such as Ruby on Rails. |
| sendgrid | - | All | SendGrid is an awesome service that helps you send large amounts of email (bells and whistles included) without spending large amounts of money. This gem allows for painless integration between ActionMailer and the SendGrid SMTP API. |
| ffaker | - | All | A library for generating fake data such as names, addresses, and phone numbers |
| chosen-rails | - | All | Integrate Chosen javascript library with Rails asset pipeline. Chosen is a library for making long, unwieldy select boxes more user friendly. |
| select2-rails | 3.4.2 | All | Integrate Select2 javascript library with Rails asset pipeline. Select2 is a jQuery based replacement for select boxes. It supports searching, remote data sets, and infinite scrolling of results. |
| sort\_alphbetical | - | All | Ruby: sort UTF8 Strings alphabetical via Enumerable extension |
| cocoon | - | All | Dynamic nested forms using jQuery made easy; works with formtastic, simple\_form or default forms |
| delayed\_job\_active\_record | - | All | ActiveRecord backend integration for DelayedJob 3.0+ |
| airbrake | - | All | This is the notifier gem for integrating apps with Airbrake. When an uncaught exception occurs, Airbrake will POST the relevant data to the Airbrake server specified in your environment. |
| whenever | - | All | Whenever is a Ruby gem that provides a clear syntax for writing and deploying cron jobs. |
| naught | - | All | A toolkit for building Null Object classes in Ruby. |
| psych | - | All | Psych is a YAML parser and emitter. Psych leverages [libyaml](http://pyyaml.org/wiki/LibYAML) for its YAML parsing and emitting capabilities. In addition to wrapping libyaml, Psych also knows how to serialize and de-serialize most Ruby objects to and from the YAML format. |
| factory\_girl | - | All | A library for setting up Ruby objects as test data. |
| listen | - | All | The Listen gem listens to file modifications and notifies you about the changes. |
| execjs | 1.4.0 | All | ExecJS lets you run JavaScript code from Ruby. It automatically picks the best runtime available to evaluate your JavaScript program, then returns the result to you as a Ruby object. |
| sass-rails | 3.2.3 | Development | This gem provides official integration for Ruby on Rails projects with the Sass stylesheet language. |
| coffee-rails | 3.2.1 | Development | CoffeeScript adapter for the Rails asset pipeline. Also adds support to use CoffeeScript to respond to JavaScript requests (use .js.coffee views). |
| uglifier | >=1.0.3 | Development | Ruby wrapper for UglifyJS JavaScript compressor. UglifyJS is a JavaScript parser, minifier, compressor or beautifier toolkit. |
| jquery-rails | - | Development | A gem to automate using jQuery with Rails 3. |
| capistrano | - | Development | Remote multi-server automation tool. |
| eycap | - | Development | Engine Yard specific capistrano recipes. The Engine Yard capistrano tasks are for use specifically with Engine Yard Managed services. But can be used as examples for building other tasks as well. |
| guard-cucumber | - | Development | Guard::Cucumber allows you to automatically run Cucumber features when files are modified. |
| guard-livereload | - | Development | LiveReload guard allows to automatically reload your browser when 'view' files are modified. |
| rack-livereload | - | Development | Bring in livereload.js into handy Rack middleware. |
| guard-ctags-bundler | - | Development | Guard-CTags-Bundler generates [ctags](http://ctags.sourceforge.net/) for your project and for gems in your bundle. For project tags file tags is generated, for gems tags file gems.tags is generated. |
| zeus | - | Development | Zeus preloads your Rails app so that your normal development tasks such as console, server, generate and specs/tests take **less than one second.** |
| awesome\_print | - | Development | Awesome Print is a Ruby library that pretty prints Ruby objects in full color exposing their internal structure with proper indentation. |
| mails\_viewer | - | Development | MailsViewer is a Rails engine to let you easily view and send mails in non-production environment. MailsViewer is a mails preview Engine which provides a table view for all mails under tmp/mails. So you can easily preview the mails in non-production mode and no longer need to worry about accidentally sending a test email to someone else’s address. |
| heroku-headless | - | All | Push from your git repo to a heroku app without any external configuration. |
| rspec | 2.14.1 | Development, Test | Behaviour Driven Development for Ruby. |
| rspec-rails | 2.14 | Development, Test | **rspec-rails** is a testing framework for Rails 3.x and 4.x. |
| guard | >=0.6.2 | Development, Test | Guard is a command line tool to easily handle events on file system modifications. |
| guard-bundler | >=0.1.3 | Development, Test | Bundler guard allows to automatically & intelligently install/update bundle when needed. |
| guard-rspec | >=0.4.3 | Development, Test | Guard::RSpec allows to automatically & intelligently launch specs when files are modified. |
| rb-fsevent | - | Development, Test | FSEvents API with signals handled (without RubyCocoa). |
| konacha | 3.0.0 | Development, Test | Test your Rails application's JavaScript with the mocha test framework and chai assertion library. |
| chai-jquery-rails | - | Development, Test | [chai-jquery](https://github.com/chaijs/chai-jquery) is a plugin for [chai](http://chaijs.com/) that gives you jquery expectation helpers.  This is a simple gem that adds chai-jquery to the rails asset pipline. |
| thin | - | Development | Tiny, fast & funny HTTP server. |
| unicorn | - | Heroku | Unicorn is an HTTP server for Rack applications designed to only serve  fast clients on low-latency, high-bandwidth connections and take  advantage of features in Unix/Unix-like kernels. |
| rspec-fire | - | Test | Checks that stubbed methods on your test double exist, but still allow you to run in isolation when you choose. A failure will be triggered if an invalid method is being stubbed. |
| fivemat | - | Test | MiniTest/RSpec/Cucumber formatter that gives each test file its own line of dots. |
| cucumber-rails | - | Test | Rails Generators for Cucumber with special support for Capybara and DatabaseCleaner. |
| capybara | 2.1.0 | Test | Acceptance test framework for web applications. |
| capybara-screenshot | - | Test | Automatically save screen shots when a Cucumber Capybara scenario fails. |
| pry-rails | - | Test | Avoid repeating yourself, use pry-rails instead of copying the initializer to every rails project. This is a small gem which causes rails console to open [pry](http://pry.github.com/). It therefore depends on pry. |
| database-cleaner | - | Test | Strategies for cleaning databases in Ruby. Can be used to ensure a clean state for testing. |
| email-spec | >=1.2.1 | Test | A collection of matchers for RSpec/MiniTest and Cucumber steps to make testing emails go smoothly. This library works with ActionMailer and Pony. When using it with ActionMailer it works with DelayedJob, ActiveRecord Mailer, and action\_mailer\_cache\_delivery. |
| launchy | >=2.1.0 | Test | Launchy is helper class for launching cross-platform applications in a fire and forget manner. |
| rspec-set | - | Test | Set is a little RSpec helper that speeds-up drastically integration tests that relies on active record objects. |
| shoulda-matchers | - | Test | Collection of testing matchers extracted from Shoulda. shoulda-matchers provides Test::Unit- and RSpec-compatible one-liners that test common Rails functionality. These tests would otherwise be much longer, more complex, and error-prone. |
| rspec-juint-formatter | - | Test | RSpec results formatted as JUnit XML that Hudson can read. |
| simplecov | - | Test | Code coverage for Ruby 1.9+ with a powerful configuration library and automatic merging of coverage across test suites. |
| timecop | - | Test | A gem providing "time travel" and "time freezing" capabilities, making it dead simple to test time-dependent code. It provides a unified method to mock Time.now, Date.today, and DateTime.now in a single call. |

## Session Handling

iPortal uses cookie based session to store session information.

In this session hash is directly stored in a cookie on a client side. The server retrieves the session hash from the cookie and eliminates the need of session id. That will greatly increase the speed of application. But taking security concerns into consideration, it is a controversial storage option.

Some advantages and disadvantages for using cookie based session store:-

Advantages

* No server storage
* No extra queries server-side
* No setup effort

Disadvantages

* User accessible/viewable, possible tampering
* 4kb size limit
* Tied to a single physical client machine
* Sent with each request(increasing total bandwidth)

Session handling is done internally by Devise gem.

*S*ession for user will begin when user successfully logins to application. The session will expired automatically if no request is received from user for 15 minutes.

## Exception Handling

iPortal uses Airbrake gem for notifying exception. When an uncaught exception occurs, Airbrake will post the relevant data to the Airbrake server specified in environment.

For iPortal application, for production environment, Airbrake server posts the exceptions to the following host :

verdacom-errbit.herokuapp.com

Airbrake will give detail summary about an error like detailed error history, file from which exception is raised, backtrace, parameters, sessions to fix etc.

Exceptions will be propagated in original form to the calling action. The calling action will handle these exceptions.

# Appendix

For Document Management data dictionary refer Appendix1.

For Safety Notification Management data dictionary refer Appendix2.

For Adaptive Monitoring Management data dictionary refer Appendix3.

****