**Project Management Solution**

Hanoi, 2014

**ProjectKit**

**Architectural Design Document**

**Bui Sy Nguyen**

SIGNATURE PAGE

AUTHOR: Bui Sy Nguyen 2014/11/07

Project Solution Architect Leader

Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** | **Description** |
| 0.1 | 2013/10/01 | Bui Sy Nguyen | Create | First created version |
| 0.9 | 2014/11/06 | Bui Sy Nguyen | Update | Pre-release version |
| 1.0 | 2015/11/23 | Bui Sy Nguyen | Update | First stable version |
|  |  |  |  |  |
|  |  |  |  |  |

# Table of Contents

[SIGNATURE PAGE 1](#_Toc419727804)

[Change History 2](#_Toc419727805)

[Table of Contents 3](#_Toc419727806)

[Definitions, Acronyms, and Abbreviations 6](#_Toc419727807)

[1 Outline 8](#_Toc419727808)

[1.1 Design Policy 8](#_Toc419727809)

[1.2 System Overview 10](#_Toc419727810)

[2 Architectural goals and constraints 11](#_Toc419727811)

[2.1 Overview 11](#_Toc419727812)

[2.2 Standards 11](#_Toc419727813)

[2.3 Integration 11](#_Toc419727814)

[2.3.1 Internal 11](#_Toc419727815)

[2.3.2 External 11](#_Toc419727816)

[2.4 Security 12](#_Toc419727817)

[2.4.1 Authentication 12](#_Toc419727818)

[2.4.2 Authorization 12](#_Toc419727819)

[2.4.3 Auditing 13](#_Toc419727820)

[2.4.4 Data Security/Privacy 13](#_Toc419727821)

[2.5 Development & Runtime Environment 14](#_Toc419727822)

[2.5.1 Development Environment 14](#_Toc419727823)

[2.5.2 Runtime Environment 15](#_Toc419727824)

[2.6 Performance 16](#_Toc419727825)

[2.6.1 Transaction and Response Time 16](#_Toc419727826)

[2.6.2 Database Volume 16](#_Toc419727827)

[2.6.3 Server Resource and Load Balancing 16](#_Toc419727828)

[2.6.4 Flex Module Load Optimizing 17](#_Toc419727829)

[2.7 Localization 18](#_Toc419727830)

[2.8 Deployment 18](#_Toc419727831)

[3 Use Case View 18](#_Toc419727832)

[3.1 Overview 18](#_Toc419727833)

[3.2 Significant Use Cases 18](#_Toc419727834)

[3.2.1 Project CRUD 18](#_Toc419727835)

[3.2.2 Project Planning 19](#_Toc419727836)

[3.2.3 Project Tracking and Monitoring 20](#_Toc419727837)

[3.2.4 Project Terminating and Project Closing 21](#_Toc419727838)

[4 Logical View 23](#_Toc419727839)

[4.1 System Layers 23](#_Toc419727840)

[4.1.1 System Logical Layer Diagram 23](#_Toc419727841)

[4.1.2 Responsibilities 24](#_Toc419727842)

[4.2 System Components List 26](#_Toc419727843)

[5 Process View 27](#_Toc419727844)

[5.1 Browser/Flash Layers 28](#_Toc419727845)

[5.2 PHP/JBoss Application Server 28](#_Toc419727846)

[5.3 JBoss Application Server for SSO 28](#_Toc419727847)

[5.4 Exchange Server and Active Directory 28](#_Toc419727848)

[5.5 Application Server for DMS 28](#_Toc419727849)

[5.6 Microsoft .NET Service for MPP Exporting (proposal) 28](#_Toc419727850)

[6 Deployment View 29](#_Toc419727851)

[6.1 Deployment Diagram 29](#_Toc419727852)

[6.2 Component Deployment Diagram 29](#_Toc419727853)

[6.3 Server Sizing 30](#_Toc419727854)

[7 Implementation View 30](#_Toc419727855)

[7.1 ProjectKit Server Implementation Diagram 30](#_Toc419727856)

[7.2 ProjectKit Client Implementation Diagram 31](#_Toc419727857)

[7.3 Scaffolding 32](#_Toc419727858)

[7.3.1 Implementation Strategy 32](#_Toc419727859)

[7.3.2 Detailed Design Policy 32](#_Toc419727860)

[7.3.3 ProjectKit Framework 32](#_Toc419727861)

[8 Quality Attributes 33](#_Toc419727862)

[8.1 Availability 33](#_Toc419727863)

[8.1.1 Availability 33](#_Toc419727864)

[8.1.2 Clustering 33](#_Toc419727865)

[8.2 Reliability 33](#_Toc419727866)

[8.3 Extensibility 34](#_Toc419727867)

[8.4 Security 34](#_Toc419727868)

[8.5 Usability 35](#_Toc419727869)

[8.6 Maintainability 35](#_Toc419727870)

[8.7 Multi-Tenant Support 35](#_Toc419727871)

[8.8 SSO 35](#_Toc419727872)

[8.9 Others 36](#_Toc419727873)

[9 Cross-cutting Concerns 36](#_Toc419727874)

[9.1 Transaction Management 36](#_Toc419727875)

[9.1.1 Hibernate Transaction Management 36](#_Toc419727876)

[9.1.2 Declarative Transaction Management, Transactional DAO and Service 36](#_Toc419727877)

[9.2 Logging and Instrumentation 36](#_Toc419727878)

[9.3 Exception Management 37](#_Toc419727879)

[9.4 Authentication and Authorization 37](#_Toc419727880)

[9.4.1 Authentication 37](#_Toc419727881)

[9.4.2 Authorization 37](#_Toc419727882)

[9.5 Caching 37](#_Toc419727883)

[9.5.1 Client Caching 37](#_Toc419727884)

[9.5.2 Server Caching 38](#_Toc419727885)

[10 Appendix 38](#_Toc419727886)

[10.1 System Design Patterns 38](#_Toc419727887)

[10.1.1 MVC, IoC and DI 38](#_Toc419727888)

[10.1.2 Yii DI 38](#_Toc419727889)

[10.1.3 Flex Client MVC Microarchitecture 38](#_Toc419727890)

[10.2 System Permissions 39](#_Toc419727891)

[10.3 Message/DTO format 39](#_Toc419727892)

[10.4 Server Sizing 39](#_Toc419727893)

[10.4.1 Peak Assumptions and Estimations 39](#_Toc419727894)

[10.4.2 Server Sizing 41](#_Toc419727895)

[10.5 Multi-Tenant Support 43](#_Toc419727896)

[10.5.1 SaaS Approach 43](#_Toc419727897)

[10.5.2 Security 43](#_Toc419727898)

[10.5.3 Extensibility 44](#_Toc419727899)

[10.5.4 Scalability 44](#_Toc419727900)

[10.6 System Configuration 44](#_Toc419727901)

[10.6.1 System configuration file list 44](#_Toc419727902)

# Definitions, Acronyms, and Abbreviations

All terms, acronyms, and abbreviations used in this document have been defined in the project’s Glossary document.

|  |  |  |
| --- | --- | --- |
| **No** | **Acronym** | **Description** |
| 1 | PK | ProjectKit system |
| 2 | TPD/H/M/S | Transactions per day/hour/minute/second |
| 3 | SRS | Software Requirement Specification/System Requirement Specification |
| 4 | DB | Database |
| 5 | XML | Extensible Markup Language |
| 6 | HTTP | Hyper Text Transfer Protocol |
| 7 | HTTPS | Hypertext Transfer Protocol over Secure Socket Layer |
| 8 | RIA | Rich Internet Application |
| 9 | Biz Flow | Business Flow |
| 10 | DTO | Data Transfer Object |
| 11 | DAO | Data Access Object |
| 12 | RMI | Remote Method Invocation |
| 13 | SaaS | Software as a Service |
| 14 | RSLs | Runtime-Shared-Libraries |
| 15 | SSO | Single Sign On |
| 16 | CAS | Central Authentication Service |
| 17 | QCD | Quality Cost Delivery |
| 18 | QnA/Q&A | Question and Answer |
| 19 | MTTF | Mean Time To Failure |
| 20 | RDBMS | Relational Database Management System |
| 21 | DI | Dependency Injection |
| 22 | IoC | Inversion of Control |
| 23 | AOP | Aspect-Oriented Programming |
| 24 | MVC | Model-View-Controller |
| 25 | WBS | Work Breakdown Structure |
| 26 | LRU | Last Recent Used |
| 27 | Spring | Spring framework |
| 28 | Hibernate | Hibernate ORM framework |
| 29 | Yii | Yii framework (PHP) |
| 30 | CRUD | Create-Read-Update-Delete |
| 31 | AS | Action Script |
| 32 | ACL | Access control lists |
| 33 | Resource | Project resource |
| 34 | SWC | An archive file of class library for Flex components and other assets |
| 35 | SWF | Shockwave Flash (file) |
| 36 | LDAP | Lightweight Directory Access Protocol |
| 37 | PM | Project Manager |
| 38 | PA | Project Administrator |
| 39 | LBS | Load Balancing Server |
| 40 | ANE | AIR Native Extension |
| 41 | AMF | Action Message Format |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Outline

## Design Policy

This section helps you know the purpose of this document, the inputs for creating this document, the design viewpoints and the structure of this design document.

***Document Purpose***

Purposes of this document are:

* Figuring-out the structure of ProjectKit system which comprises its components, the externally basically visible properties of those components, and the relationships between them.
* Describing the ProjectKit system from the viewpoint of different stakeholders, such as end-users, developers and project managers
* Supplying not only a set of architectural patterns and design principles and also a high-level design of ProjectKit system to

***Design Inputs***

The inputs for creating this document are:

* ProjectKit System’s functional requirement specification created by Requirement Team.

Refer to SRS documents folder.

* ProjectKit System’s non-functional created by Requirement Team, refer to:

Non Functional Requirement.xls

* The decision of using technologies and tools to develop the ProjectKit system:

1. Develop a web system using RIA technology
2. Use Flex at RIA client side

***Design Viewpoints***

This architecture design is created by these viewpoints:

* ***Architecture Model:***

This document provides a comprehensive architectural overview of the ProjectKit 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.

In order to depict the software as accurately as possible, the structure of this document is based on the “4+1” model view of architecture.



*Architecture Model Tool:* We will use *UML* as main model language for representing the architecture design.

* ***Framework-specified Design Model:*** The design model also follows the specification of the design patterns defined in section 4.2. System Design Patterns.
* ***Top-down approach:*** Depict the design from overview of system to the details of each sub system. Hence, we will get a design structure as a tree with the following node-levels:

1. *Node Level 1*: The overall ProjectKit system.
2. *Node Level 2*: The *system functions* (or the system *components*, the *sub-systems*), e.g. *Project, Requirement, Deliverable..*. We will call components at this level are *top-level components*.
3. *Node Level 3*: The sub-divisions of the function at level 2 mentioned above. We will call components at this level are *second-level components*.

* ***Design Scope:*** This document will depict the design of the system and sub-systems down to *top-level components*. The design of *second-level component* and more detailed component will be written in *detailed design document*.

***Document Structure***

The structure of this architecture design follows the 4+1 View Model’s structure; it mainly contains following 5 items (that we call *Views*):

* [***Use Case View:*** Describes the set of scenarios and/or use cases that represent some significant, central functionalities of the system.](#_Use_Case_View_2)
* [***Logical View:*** Depicts the main behavioral requirements and shows how the system is decomposed into a set of abstractions. System *top-level components* are the main elements studied in this view. In this view, we will use *layer diagram, component list* and *component communication diagram*.](#_Logical_View_1)

Purpose of this view is to help Designers in developing Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.

* [***Process View***: Describes and study the ProjectKit system's processes and how they communicate.](#_Development_View)
* [***Deployment View*** *(or* ***Physical View****)****:*** Describes how the ProjectKit application(s) is deployed on target environments (ProjectKit servers and client environment) and how they execute in a network of computers using *deployment diagram*.](#_Deployment_View)
* [***Implementation View*** *(or* ***Development View****)****:*** Describes the modules of the system using *package diagram*.](#_Implementation_View_1)

In addition, we will also study these more 4 sections:

* [***Quality Attributes and Non-Functional Design:*** Describes the system quality attributes and non-functional design items.](#_Quality_Attributes)
* [***Cross-cutting Concerns:*** Describes all cross-cutting concern of ProjectKit system in order to easily implement it by using the design patterns defined in section 10.1. System Design Patterns.](#_Project)
* [***Scaffolding:*** Describes an implementation structure of ProjectKit system using the specifications of design patterns defined in section 10.1. System Design Patterns.](#_Scaffold)
* [***Appendix:*** Section of appendices.](#_Appendix)

## System Overview

ProjectKit is a generic purpose project management system using rich internet application (RIA) technology, built on several technologies: HTML5, Apache Flex, PHP, Scala and Java.

The following diagram describes the function overview of ProjectKit System.



Picture ‑ System Functional Overview

As you see, all the functions of ProjectKit system are divided into 8 function groups. Generally, the function group from No.1 to No.7 can be accessed from any normal user in ProjectKit system. However, in some particular function, there may have some use case that can only be accessed by particular actors (PM, Owner, Project Administrator…).

The function group *No.8 - Administration* can be accessed only by *Project Administrator* or *System Administrator*.

For more details, please see in section 3. Use Case View.

|  |  |  |
| --- | --- | --- |
| **No** | **Function Name** | **Description** |
| 1 | Project | * Project CRUD operation (create, read, update general information, delete tentative) * Project List and Project Dashboard |
| 2 | Task | * Gantt Chart task-based project schedule management |
| 3 | Resource | * Project Resource management. |
| 4 | PJ Network | * My Home timeline * Project collaboration using social networking paradigm |
| 5 | Issue | * Issue management |
| 6 | Timesheet | * Timesheet input, view, approve, reject, report |
| 7 | Report | * Reporting function |
| 8 | Administration | * Project administration and system administration |

# Architectural goals and constraints

## Overview

This section describes the constraints and objectives that have some significant impact on the architecture.

## Standards

***JSON and AMF3:*** ProjectKit System use JSON and AMF for remote data transferring between ProjectKit Client and ProjectKit Server services.

* In case of integration between AS-based client and ProjectKit server, the message format will be AMF (Action Message Format) version 3.
* In case of integration between other type of client and ProjectKit server, or between ProjectKit system and other system using web service, the message format will be JSON (JavaScript Object Notation). For more details on the JSON standards see:

<http://en.wikipedia.org/wiki/JSON>.

Support for these standards is provided and shall be achieved primarily through the use of ZendAMF and BlazeDS framework.

Note that the AMF is several times faster than the JSON standard in transferring.

***MSPDI:*** (Microsoft Project Data Interchange) - Some function related to import/export MPP file will use MSPDI (that actually is XML) format.

## Integration

### Internal

***Single Sign-on Server***

Need a solution for SSO integration here!

***Notification Push***

Need a solution for Notification Push integration here!

### External

***PHP Server***

The ProjectKit System uses PHP as front-end application server. Data communication between Flex client application and PHP front-end server is enabled by using ZendAMF+AMF3 or web services+JSON.

***Microsoft Exchange Server***

Can ProjectKit system authenticate using MS Exchange account?

***JBoss Application Server***

JBoss Application Server is the server used for deploying, hosting web applications including ProjectKit Server application, CAS Server application and other tenant application.

The ProjectKit System may do some communications to this server when using JNDI to lookup database connections.

***Apache Server (Httpd Server)***

Apache Server or Httpd Server is used for load balancing and clustering solution.

## Security

### Authentication

Need a single sign on (SSO) solution here!

### Authorization

ProjectKit system uses role-based approach for authorization. This allows user can only access to the ProjectKit functions that are defined for his assigned role.

For example, a user with *PM role* cannot create a project but can create the master schedule for a given project, while a user with *PA* role can create a new project.

The following diagram depicts the authorization flow of ProjectKit



Picture ‑ ProjectKit System Authorizing Diagram

The authorization is performed under several scopes including module, screen and operation, by using Yii Authorization both in PHP webpages and Flex client. At each scope, the authorizing process will check the user role and the ProjectKit security matrix which is a table of role-right records to determine whether the user can or cannot do a certain operation.

### Auditing

The system will collect auditing data related to changes to key data elements. For example, the important entities such as *Resource*, *Project*, *Task*, *JobRecord*, *Timesheet* always have the auditing fields: *createUserId*, *createTime*, *updateUserId*, *updateTime*. See the use case documentation for data elements for which audit tracking is required.

The auditing solution of ProjectKit System also includes auditing log. This log contains records showing who has accessed our system and what operations he has performed during a given period of time. The details of that auditing time period is defined later, for example, it can be daily, weekly, monthly or yearly…

The system administrator is in-charge to audit ProjectKit system and back-up ProjectKit audit data including DB audit data and audit log files. ProjectKit system also supplies several back-end tools for analyzing/reporting DB audit data and audit log files. For example, *the Reliability measuring tool* is used for measure reliability of system.

### Data Security/Privacy

***Security and Privacy Policy***

Data security shall follow the corporate data security guidelines. In the absence of clearly defined corporate guidelines, current industry best practices should be followed.

Special consideration will be taken with sensitive data elements such as passwords and personal data such as social security number and employer identification numbers. Password data will be protected using hash and salt values. Identification numbers shall be collected only when necessary and viewing shall be limited to only the users who have need.

ProjectKit system will also use data privacy that is the process of securing critical data assets such as credit cards, social security numbers, passwords, and addresses as they are being stored, transmitted, and used within the enterprise.

***Secure AMF Encryption***

The data transferred between ProjectKit Client and ProjectKit Server will be encrypted by this standard. Support for this standard is provided and shall be achieved through the use of ZendAMF or BlazeDS framework.

***SSL/Https***

The client login form data and all data transferred between ProjectKit system and other systems through web services will use this standard for encrypting.

***Storage Data Encryption***

The critical data assets such as credit cards, social security numbers, passwords, and addresses will be encrypt when they are stored in database of ProjectKit System. The encrypting standard for these data is 256-bit AES encryption, and the AES implementation should be provided by a public vendor and be carefully reviewed.

***Client Data Encryption***

The ProjectKit Client SWF file should be purged in order to limit the ability of hackers in attempting to decompile it.

## Development & Runtime Environment

### Development Environment

The following table shows the items of development environment including OS, middleware, software and hardware.

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | Operating System | *Developer’s OS:*  *DB Server:*  *Application Server (from IT stage):*  *SSO Server:*  *MPP Import-Export Server:* |
| 2 | Middleware | *RDBMS:* MySQL 5.5+  *Application Server:*  WAMP server with PHP 5.4+  JBoss AS 7.1+/ JBoss EAP 6.1+ |
| 3 | Software | Yii 1.1.12+  JDK 1.7+  BlazeDS 3.2+  JUnit  Flex SDK 4.9+  FlexUnit  Microsoft Project 2010  Microsoft Office 2010  Microsoft IE version 7.0+  Adobe Flash Player 11.3+  Mozilla Firefox 3.0+ |
| 4 | Hardware | *TBD* |
| 5 | Development language and Tool | *Development Language:*  Java  Scala  PHP 5.4+  Action Script 3.0  JavaScript/CSS3.0  *Development Tool:*  JetBrains PHP Storm 7.0+  JDK 1.7+  Eclipse 3.7+  Git (**bitbucket.org** and **github.com**)  Flex SDK 4.9+  Adobe Flash Builder 4.7 |

For more details about the combination of these software and middleware, see 4.1.System Layers.

### Runtime Environment

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | Operating System | TBD |
| 2 | Middleware | *RDBMS:* MySQL 5.5+  *Application Server:*  CentOS 6.x  LAMP/WAMP Server 2.2 with PHP 5.3+  JBoss AS 7.1+/ JBoss EAP 6.1+  *Clustering and Load Balancing middleware:*  Apache 2.2.22+ |
| 3 | Software | PHP 5.4+  JRE 1.7+  BlazeDS 3.2+  Flex SDK 4.9+  Microsoft Project 2003  Microsoft Office 2007  Microsoft IE version 7.0+  Adobe Flash Player 11.3+  Mozilla Firefox 3.0+ |
| 4 | Hardware | See 6. Deployment View for more details |

For more details about the combination of these software and middleware, see 4.1.System Layers.

## Performance

### Transaction and Response Time

* Average transaction rate: ? TPD
* Maximum concurrent users: ?
* Average concurrent users: ?

### Database Volume

* Number of available users: ?
* Number of project: ?
* Average number of timesheet records/project: ?
* Average number of defect records/project: ?
* Average number of task records/project schedule: ?

### Server Resource and Load Balancing( Nghiên cứu

* Server resource: is there any requirement for server resource?
* Load balancing:

What tool will be used for load balancing?



Picture ‑Load Balancing Diagram

### Flex Module Load Optimizing

Use Flex’s Runtime-Shared-Libraries (RSLs) mechanism to do this.

Modules are SWF files that can be loaded and unloaded by an application. They cannot be run independently of an application, but any number of components can share the modules. Modules let you split your application into several pieces. ProjectKit client application can dynamically load other modules that it requires, when it needs them. It does not have to load all modules when it starts, nor does it have to load any modules if the user does not interact with them. When the application no longer needs a module, it can unload the module to free up memory and resources.

* Smaller initial download size of the SWF file.
* Shorter load time due to smaller SWF file size.

See 7.2.ProjectKit Client Implementation View for more details.

## Localization

ProjectKit system’s user interfaces use English as main language. It also has a localization support for other languages like Japanese and Vietnamese.

## Deployment

Although the almost application users are intranet users, in some cases, internet users from several countries will also be accessing parts of the system.

It is also assumed that remote users will have high speed internet access but will not likely be a connection as fast as a typical 100Mbps LAN. A low maintenance deployment strategy is required to support this diversity in locations and access speeds.

# Use Case View

## Overview

This section depicts use cases which are significant with respect to the architecture of the system. The selected use cases represent at a high level the flow of orders processed through the application.

See the use case specification documents and business workflow documents for the complete set of use-cases that are defined for the application.

## Significant Use Cases

### Project CRUD

To start using ProjectKit system, we must first create a project. A user with PA (Project Administrator) role can create a project and a PM (Project Manager) can update the general information of the created project.

Any user with Project Member role can see the project information.



Picture ‑Use Case Diagram: Project CRUD

### Project Planning

After a project is created, the PM can start planning and tracking activities by using *Project Planning* function.

This function allows all users of project can create a task-based schedule with *Gantt chart* that is a set of *WBS tasks*.

The *Resource Management* function supports PM can do resource planning and quickly find a resource appropriate with a task.

The *Issue Management* function supports managing project’s changes, risks and issues.



Picture ‑Use Case Diagram: Project Planning

### Project Tracking and Monitoring

After planning phase, users input actual project data for ProjectKit system. These data include:

* Actual progress of each task assignment in project WBS schedule. The progress is directly input into Gantt chart.
* Actual effort used for each task, this effort is input by Timesheet function, which allows users declare the actual effort of their tasks every day.
* Defects which are input by DMS sub-system. The project quality tracking allows users view the quality status of project.
* Deliverable progress.
* Requirement progress.
* Risk/Issue escalation.
* Performance Assessments.



Picture ‑ Use Case Diagram: Project Tracking and Monitoring

### Project Terminating and Project Closing

For some reason, we need to cancel (terminate) a project that is running. The project lifetime may be terminated at Project Terminating use case.

In case of a project has done all it necessary lifetime activities, it can be closed by using Project Close use case.



Picture ‑Project Terminating and Project Closing

# Logical View

## System Layers

### System Logical Layer Diagram

MySQL

AD/LDAP

Database

NoSQL

Data  
Access

Active Record

Active Record / Hibernate Transactions / NoSQL Operations

Service

Active Service

Active Controller

ZendAMF/BlazeDS/REST

Rich Internet  
Presentation

ProjectKit GUI Flash Player 11.3+ / AIR 3.5+ / HTML5

ActionScript 3.0

Flex SDK 4.0+

Java DAO

NoSQL Data Access

External Systems

External API

Service Consumers

Users



4.1.1‑ System Software Stack

The diagram above represents the basic software, middleware runtime libraries and OS taking part in ProjectKit system. Note that in actual runtime environment, there may have more software like SSO Client, Spring Security, log4j… which are not mentioned here. To see how layers work and what their responsibilities are, let’s continue to next section.

* *Users/Interface Layer*: Contains human-activity of user or interface of Internal/External System/Subsystem to interact with ProjectKit system.
* *Rich Internet Presentation* layer: Contains ProjectKit graphic user interfaces. See 10.1.3 Flex Client MVC Microarchitecture for more details
* *Service* layer: Contains ProjectKit services components. These components work under RESTful ActiveController, AMF ActiveService or BlazeDS service components.
* *Data Access* layer: Contains ProjectKit Data Access like custom DAO or extended Yii’s ActiveRecord components, working on coordination of Hibernate or Yii’s ActiveRecord transactions.
* *Database* layer: Data storage of ProjectKit system.

### Responsibilities

***Components Stack***



Picture ‑1- System Component Stack

In the diagram above, we can see how the components are physically distributed in logical layers.

When user interacts to a screen of ProjectKit GUI layer, data is pre-processed at client side (i.e. client validation) and stored into a DTO object that is instance of a Dto class written in AS3. The *service proxy object* will invoke an appropriate *remote method* from *service component* that is hosting on server side.

At server side, the BlazeDS Massage Broker servlet will extract the data from AMF data to a Java Dto object and invoke the calling service method with the extracted data. The service component will execute business logic of the method, and there may have an invocation to DAO component(s). The DAO component and then may execute some data statement through Hibernate 3 transactions.

After all, the data is stored to ProjectKit Database.

From database, the result data will be returned to ProjectKit Client just in that way, but by inverse direction.

***Business and Logic Distribution to Layers***

There’re many programming business and logics we must implement to build ProjectKit system. The following image shows where that business and logics should be located; which layer is that one business or logic should belong to:



Picture ‑2- Business Distribution for implementation of ProjectKit System

|  |  |  |
| --- | --- | --- |
| **No** | **Function Name** | **Description** |
| 1 | Server logging | Logging at server side (using Yii DI) |
| 2 | DAO transaction | Data Transaction business logic (using Yii DI) |
| 3 | DAO component | Data Access interfaces and implementation classes. This DAO contains the basic CRUD data manipulations |
| 4 | Service transaction | Service Transaction business logic (using Yii DI) |
| 5 | Business service component | Business service interfaces and business service implementation classes. These are the most important components contains the implementation of almost all business rule and complicated data manipulations. These services will be exposed for remote accessing at client side. |
| 6 | Server validation | Business validation at server side |
| 7 | Client logging | Logging at client side (using AS3) |
| 8 | Service remote invocation | Remote invocation to remote service components (mentioned at No.5) |
| 9 | Service proxy interface | The proxy interface written in AS3 for remote service component |
| 10 | Screen visualization | The visualization components for data input, i.e. the *Gantt Chart*, *Resource Sheet* components. |
| 11 | Screen transition | The screen transition business logic |
| 12 | Client validation | Input validation at client side |
| 13 | Input method | All other input support logics at client side |

## System Components List

The following list enumerates the *top-level components* of ProjectKit System. Each of these components may consists of several GUI component (a GUI component at top-level is just a screen) and controller component (i.e. the service components). The details of these components are described in detailed design document.

|  |  |  |
| --- | --- | --- |
| **No** | **Component Name** | **Description** |
| 1 | Project | The component used for project CRUD, Project List and project Details |
| 2 | Task | The component used for creating Gantt chart, task assignment and tracking project progress |
| 3 | Resource | The component used for human resource management activities. |
| 4 | Issue | [Need to document] |
| 5 | Timesheet | The component used for timesheet management |
| 6 | JobRecord | [Need to document] |
| 7 | Report | [Need to document] |
| 8 | Administration | [Need to document] |

# Process View

The system is composed of several processes. Many of these expose interfaces through which external entities can interact with it. There are active BlazeDS Servlet, DMS Servlet, Central Authentication Service (CAS), Flex Client Application, Exchange Server, Active Directory, MPP Export Web Service.

The details of Database Server, Httpd Server, Exchange Server and Active Directory are not discussed here because they are not custom components of the system. See the list of applications that require integration in section 2.3. Integration.

The following diagram depicts the process view of ProjectKit system:



Picture ‑The Process View of ProjectKit System

The process view includes several *threads* and *processes*. Each process-chain contains processes and threads, and always belongs to at least one root process. Data transferred from one thread cannot directly share to other thread as they can do inside the thread.

## Browser/Flash Layers

This process contains the thread of *ProjectKit Client* that manages all business at client side, including data process, visualization, screen transition, remote method invoking.

In real situation, you can imagine that each time one user login to ProjectKit system from a browser, one thread instance of the *ProjectKit Client* will be initialized. That thread instance controls the following business:

*ProjectKit GUI*: the business for data input, screen visualization, screen validation

*ProjectKit Service Proxy*: the business for remote method invocation

## PHP/JBoss Application Server

This is the process of ProjectKit Server application. It is hosted and managed by PHP or JBoss Application Server.

This process contains the *BlazeDS Message Broker Servlet* that is Java servlet thread for AMF/JSON message broking.

Each time a client request arrived, the servlet will create a thread instance to serve the request. This BlazeDS servlet thread will extract data from request and send it to ProjectKit’s service beans.

The service bean and then does the business operations. At this time, there may have some data transition needs, the *Hibernate Entity Manager* will be invoked to do some SQL statements to satisfy that needs, and then the *Entity POJOs* is used to organize return query data and pass it to beans receive as query result.

In case of multi-tenant, each tenant will be deployed in an application server. There may be several tenant deployed in one application server. Each time the server starts, it creates one process for one tenant.

## JBoss Application Server for SSO

This is process the process of Single Sign-on service. It is hosted and managed by PHP or JBoss Application Server.

See section 2.3.1 for more details.

## Exchange Server and Active Directory

Microsoft Exchange Server and Active Directory will provide service interfaces for:

* Sending/receiving mail
* Active directory information retrieving for user authenticating.

The process of these services is managed by Microsoft Windows Server.

## Application Server for DMS

This is the process of external system that serves the request of ProjectKit system for creating DMS record.

## Microsoft .NET Service for MPP Exporting (proposal)

*(This is proposed solution for MPP Export function that enables exporting schedule data to binary MPP file)*

MPP Export Service is the web service developed under Microsoft Communication Foundation (WCF) and deployed as a Microsoft Active Server Page (ASP.NET) application.

This .NET application uses WebORB for .NET as the remoting gateway. It also uses MPXJ .NET to handle Microsoft Project data.

# Deployment View

## Deployment Diagram



Picture – System Deployment Diagram

The diagram above depicts the deployment view of ProjectKit system.

Note that at *Balancing/Clustering server*, we may use *nginx* or *mod\_cluster*. Both of them are open source and can use for *clustering* and *load balancing*.

## Component Deployment Diagram

The following diagram depicts where components from the various logical layers of the application will physically reside. Note that the components depicted often represent a category of components. The actual application will have more components than depicted here.



Picture ‑ Component Deployment Diagram

## Server Sizing

The sever sizing is describe at section 10.4

# Implementation View

## ProjectKit Server Implementation Diagram

The following diagram depicts the implementation view (or *development* view) of ProjectKit system at server side:

‑ProjectKit Server Implementation Diagram

* ***ProjectKit.war***: web application archive file for ProjectKit Server application.
* ***configuration***: web folder(s) contains system configuration files.
* ***service***: package that contains business service component’s interfaces
* ***service.impl***: package that contains business service component’s implementation classes
* ***dao***: package that contains DAO component’s interfaces
* ***dao.impl***: package that contains DAO component’s implementation classes
* ***common***: package contains common classes and interface such as *DTOs*, *entities*, *utility classes*…

## ProjectKit Client Implementation Diagram

The following diagram depicts the implementation view of ProjectKit system at client side:



Picture ‑ ProjectKit Client Implementation Diagram

The client application is physically distributed into several SWC and SWF files. Each of these files is corresponding to a system component, and should be designed or **implemented** as a separated module or separated library project.

In order to improve load performance of client application, each RSL or module must have the physical size *not exceed 200KB*.

These modules are divided into these categories:

* Function module: each function component have its own module: Project, Task, Issue …
* Non-function module: common module

At logical implementation view, in each module there are several packages are included. These packages are divided into these categories:

* Visualization: the ProjectKit common screen components: GanttChart control, AdvancedTree control…
* Common Model: the common *model* in MVC framework.
* Core: contains the abstract and base component classes.
* View: the business-specified view component of each function.
* Service: contains all remote service proxy class and client-side service management components.

## Scaffolding

### Implementation Strategy

***Implementation Process***

The process for implementing ProjectKit system from ADD to an output component, including 3 steps:

* Complete ADD
* Develop Scaffold (or in other word, *scaffolding*)
* Develop Components

***Implementation Work Break Down Policy***

There’re 2 ways to break down the implementation:

* Break down by components: We will have 14 task groups to create 14 components
* Break down by Client/Server: We will have 2 task groups: ProjectKit Server (java component) and ProjectKit Client (Flex components)

### Detailed Design Policy

The detailed design policy will be developed before we create Detailed Design Document. This policy includes:

* Detailed design template
* Detailed design sample
* Detailed design review checklist

### ProjectKit Framework

The ProjectKit Frame work includes:

* Scaffold: The scaffold is the development directory with all necessary common files created at *scaffolding* step in *implementation process* mentioned in section 7.3.1. These files include:
* Tool-generated source files: All Entities and core DAO components. All core service and all presentation class (in AS3) for CRUD operations with each entity.
* System configuration files (see Appendix 10.6)
* Common Library, includes:

*Server Utils:* DateTimeUtils, ConvertUtils, StringUtils

*Server Security implementation:* Data encryption.

*Authentication/Authorization:* ProjectKit UserAPI(use at CAS server), ProjectKit UserAPI client (use at AP server), Security Matrix

*ProjectKit-core:* The micro MVC framework.

*Client Utils:* DateTimeUtils, ConvertUtils, StringUtils

*Client Security implementation:* Data encryption.

*Client Common control (Gantt-chart, calendar, treeview…)*

*Caching implementation*

The details of common library design are mentioned in other document.

* Framework guideline: Guideline for design and coding an component following the specifications of Spring framework, ProjectKit common library and all design patterns mentioned in section 10.1)

# Quality Attributes

## Availability

### Availability

The system availability is measured by the following formula:

Availability = 

We set the target of Availability is 99.9%:

Availability ≥ 0.999

99.9% availability is required during business hours as the majority of use is currently via internal users. The application is also expected to be generally available at all hours of the day to internet users and intranet users working in off-hours. Exceptions are allowed for brief (< 4 hour) individually scheduled maintenance periods. These must be in off-hours and only as required to accommodate special maintenance needs. Examples would be the replacement of failed redundant hardware or server upgrades. The system shall not be designed to require regularly scheduled maintenance.

### Clustering

Use *clustering support* of JBoss Application Server (JBoss AS). This permits shutting down one server to maintain, upgrade hardware without seriously impact to the availability of other servers in the system.

## Reliability

The system architecture must provide zero tolerance for loss of data. Support for clustering is available to provide a high level of reliability and availability. By definition clusters are highly available platforms because as long as at least one of the servers in the cluster is performing its tasks the application remains available.

* Using Mean-Time-To-Failure method to measure the reliability of system:

MTTF = E[tF] = 

Where:

tF is random variable representing the *time* at which the system *first* *fails*. E[tF] is mathematical mean expectation value of tF.

R(t) is function representing the system *reliability* at the time *t,* we can obtain *R(t)* value at given value of *t* bythe following conditional probability expression:

R(t) = Pr[tF > t | initial system state]

In ProjectKit system, we set the target of MTTF is 99.9%

MTTF ≥ 0.999

*Refer*: Performance-Related Reliability Measures.pdf

For implementation, we have to use the *Java-based Reliability Library* (*JReliability*), an open source library that helps us to calculate reliability of the system.

We also have to implement logging mechanism to log exceptions of ProjectKit system in a certain format so that this tool can analyze. See 9.2 for more details.

*Refer:* <http://jreliability.sourceforge.net/>

* Clustering mechanism (see 8.1.2 Clustering) allows shutting down one server to maintain, upgrade hardware without seriously impact to the reliability of other servers in the system

## Extensibility

The system architecture must provide mechanisms to configure and extend the system in various ways so that anticipated future business needs can be accommodated without requiring significant modification or programming.

* The original supported RDBMS is MySQL, but the design also supports porting from MySQL to others RDBMS: Oracle 11+, MSSQL 2005+ with least effort.

In order to do so, all queries and SQL statements must be placed in XML files separated from source code files.

* The system components must be designed as independently as possible. The dependency between components should be placed in XML using DI (Dependency Injection), AOP (Aspect-Oriented Programming) and IoC (Inversion of Control) design pattern. For more details, see 10.1.System Design Patterns.
* Support tenant data extension, see 10.5.3. Extensibility for more details.
* The following features are identified as requiring extensibility support in the architecture:

i) *Project Management Object Model Format*: this is XML format that allows import/export a task-based project scheduled in to XML. In future, ProjectKit system can work off-line for the *project scheduling* feature.

ii) The *Gantt chart* should be designed and implemented in order to version-up to a function that can fully support scheduling method like a “Microsoft Project” in Flex 4.

* Hardware extensibility: The system can be deployed in a flexible strategy in order to easily extend hardware ability such as hard disk volume-up, memory and CPU upgrade, application server addition, httpd server addition.

## Security

The following list figures out some common security vulnerabilities to prevent:

* SQL Injection
* Forced Browsing
* Cross Site Scripting (XSS)
* Isolate Parameter
* Client Side Comment
* OS Command Injection (OS Commanding)
* Path Traversal (Directory Traversal)
* Back Door and Debug Option
* Session Hijacking
* Error Codes
* Cross Site Request Forgery (XSRF or CSRF)

The details of vulnerability and the appropriate solution are written in other documents, please refer to:

[10.5.2. Security](#_Security)

ProjectKit\_Security Solutions.xls

ProjectKit\_Security Code Review Viewpoint.xls

## Usability

* Support 4 browsers: IE7+, Firefox3.0+, Safari, Google Chrome

## Maintainability

Application interfaces with which external entities interact shall be created using a message based service oriented design. These interfaces shall not be platform dependent. Versioning of the interfaces shall also be supported and plan for deprecation of old versions of the interfaces shall be in place.

Application components shall be constructed with current best practices and documented features of the frameworks to increase maintainability of the application and to help improve compatibility when migrating it to future versions of frameworks.

The application shall comply, where feasible, with corporate application development guidelines and tools as well as with the corporate data security policy.

* ProjectKit system is designed by using preeminent design patterns which allows us easily maintain it later. See 10.1.System Design Patterns for more details.
* The design policy and coding policy must help increasing system’s modularity, independency so that the system can be easily maintained later.
* The implementation and deployment strategy must help system administrators easily audit and maintain database, maintain log information, maintain hardware and version-up the libraries/middleware.

## Multi-Tenant Support

ProjectKit system should be designed for easily upgrade to SaaS version. In case of upgrading to SaaS version, the multi-tenant support of the SaaS approach and it goals is depicted in appendix section 10.5.

## SSO

The basic flow of SSO Solution is shown as:



Picture ‑Basic flow of SSO Solution

SSO Solution is composited of 4 parts:

* Apache Server: Http and load balancing server
* Auth Server: authentication management server. See 6. Deployment View for more information.
* Application / Service API server: Application-side (client-side of SSO solution) server. For more details, please refer to the detailed document:

ProjectKit\_SSO Integrated Solution.docx

## Others

* Using UTF-8 character set for all text-based documents and source code files.

# Cross-cutting Concerns

## Transaction Management

### Yii Transaction Management

***Global Transaction***: All transactions related to multi-tenant shared data will be under global transaction.

***Local Transaction***: All business transactions of ProjectKit system will be under local transaction.

***External Transaction***: ProjectKit system does not manage transactions related to operations that use external interfaces, for example:

* A LDAP operation cannot be rolled-back if an exception occurs at AD Server.
* When sending email using MS Exchange server, if the sending operation cause exception at server-side, the ProjectKit system does not roll-back anything.

### Declarative Transaction Management, Transactional DAO and Service

***Declarative Transaction Management***: In ProjectKit system, all transactions regardless of the layer that the source code belongs to will be managed by *Declarative Transaction Management*. That is, Yii DI will read the transaction rules written in configuration file and do the necessary transaction management operations for all the source code that matches the rules. These rules are declared in Yii DI’s *transactional advice*:

* Matching rule: defines patterns of method name, class name or package name that the source code belongs to. If the running transactional source code matches this rule, the transaction will be performed.
* Rollback rule: defines patterns of Exception class name. If the running transactional source code raises this type of Exception, the rollback will be performed.

## Logging and Instrumentation

There’re 3 types of log as follows:

In ProjectKit system, logging is performed in both application server-side and client side:

* Server side logging: Operation log, Event log, Performance log
* Client side: Logging in client side by AS3 code. This log will be used for client bug tracking.

We also use Yii DI to implement logging mechanism by defining logging *aspects* and logging *join point* together with logging *point cut*.

|  |  |  |
| --- | --- | --- |
| **No** | **Log type** | **Log content** |
| 1 | Operation log | * Log level (Information, Warning, Error, Fatal error) * Error message, DateTime (millisecond) * Stack trace (place, method, line number) |
| 2 | Event log | * Log level (Information, Warning, Error, Fatal error) * Error message, DateTime (millisecond) * Stack trace (place, method, line number) |
| 3 | Performance log | * Notify Start/End process * Time stamp (millisecond) |
| 4 | Client log | * Logging in client side. This log will be used for client bug tracking. |

Table ‑Log type description

The details of Logging and Instrumentation are mentioned in the Development Guideline document.

## Exception Management

Error/Exception handling policy:

* Using common error screen component of ProjectKit flex framework to display error message.
* When an exception is caught or an error occurred (abnormal cases), error information is stored into a Global Error Stack object so that the latest exceptions can be retrieved anywhere. The maximum number of exceptions that the system can store is defined in the ERROR\_STACK\_SIZE constant.
* Do not throw any exception to upper business coding process.
* Perform logging instantly from where error occurred.

## Authentication and Authorization

### Authentication

ProjectKit allow user login using both OAuth and ProjectKit’s Authentication service.

Both of the ProjectKit web client application (Flex application) and ProjectKit Database are hosted in Tenant Servers. The authentication between ProjectKit central server and Tenant Servers is performed under SSO mechanism.

Refer to 8.8.SSO for more details.

### Authorization

ProjectKit system authorizes user by his role on each application/component that takes part in the system and by the tenant that he is belong to.

Thus, the permission of a user for accessing to a specified screen in the system depends on his role and his tenant’s role.

## Caching

ProjectKit system uses these cache solutions:

### Client Caching

* *LRU Cache*: Caching component instances using Last Recent Used algorithm. This caching solution is used in processing a large amount of data object, for example we need to cache the working time of a calendar for fast retrieving.
* *Instance Cache*: Caching component instances. This caching solution is used in processing frequently the data objects that take times in initializing.
* *Renderer Cache*: Caching the Flex item renderer’s instances for optimizing performance of presentation layer.
* *Instance Pool*: component instances pool for caching
* *Renderer Pool*: item renderer’s instances pool for caching

### Server Caching

* PHP default opcode cache
* Data Cache

# Appendix

## System Design Patterns

In order to understand the way we design ProjectKit system, we should study all design patterns applied for this design and all further detailed design of the system.

### MVC, IoC and DI

ProjectKit Client system is constructed on Flex Framework which is a preeminent MVC framework.

The *F-Project Framework* supporting IoC and DI design patterns.

All Server Java components and sub-components of our system are designed by IoC and DI patterns.

### Yii DI

For all Server Java components, we use AOP paradigm implemented in Spring framework to increase system’s modularity by allowing the separation of cross-cutting concerns.

For more details see 9.Cross-cutting Concerns.

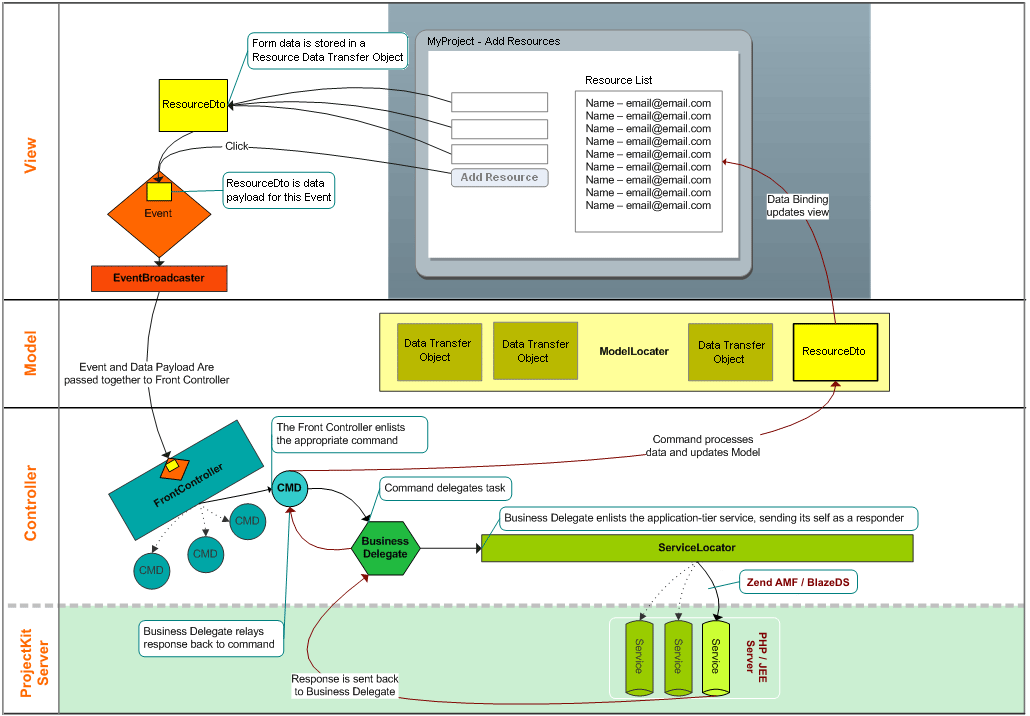
### Flex Client MVC Microarchitecture

The Presentation Layer is designed using MVC (Model-View-Controller) design pattern, and developed using Flex ActionScript 3.

*Model*: Represents the data for the application.

*View*: Renders the model in a human readable form called the User Interface. Flex SDK 4 allows you quickly create GUI component by using MXML forms together with AS3.

*Controller*: Holds the business logic that is responsible for changing the data help in the model



10.1.3‑Flex Client MVC Microarchitecture

## System Permissions

This section will be updated later.

## Message/DTO format

This section will be updated later.

## Server Sizing

This section supplies the estimations of server size that may be used for deploying ProjectKit system. Some information about server capacity and server sizing method can be found at these reference links:

<http://www.dell.com/content/topics/global.aspx/power/en/ps3q01_graham?c=us&l=en&cs=555>

<http://www.sizinglounge.com/>

<http://www-304.ibm.com/partnerworld/wps/sizing/portal/search.jsp>

### Peak Assumptions and Estimations

***Parameter Assumptions and Estimations***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Formula** | **Value** | **Unit** | **Description** |
| Nspeed = |  | 1,000 | Mbps | Network speed (megabits per second) |
| MaxTraffic = | 1024\*1024\*Nspeed/8 | 131,072 | KBps | Maximum traffic ability (kilobytes per second) |
| Sreq = |  | 1,024 | byte | The average size of each request in bytes (assumption is ~1KB) |
| Sres = |  | 10,240 | byte | The average size of each application server response in bytes (assumption is ~10KB) |
| Tp = |  | 6,000 | tph | Peak transaction rate (transactions per hour or hits per hour). Assume that all 6000 transactions of the day are performed within 01 hour. |
| TpBps = | (Sreq+Sres)\*Tp/3600 | 1,8773.33 | Bps | Transaction rate in bytes per second for peak interval of 01 hour |
| UEnet = | TpBps/MaxTraffic | 0.143% | % | Estimated value for Utilization of Network |
| TMemSize = |  | 20,480 | byte | Average memory used for one transaction (assumption is ~20KB) |
| ULnet = |  | 70% | % | Network Utilization Limit. Should always preserve 30% for network overhead |
| ULcpu = |  | 80% | % | CPU Utilization Limit |
| ULmem = |  | 90% | % | Memory Utilization Limit |
| ULdisk = |  | 85% | % | Disk Utilization Limit |
| MaxCU = |  | 1,500 | user | Peak Concurrent Users |
| UMemSize = |  | 1,706,667 | byte | Average memory used for one concurrent user including dynamic content (assumption is ~1MB) |
| APMemInit = |  | 300 | MB | Initialized memory used at AP server. This is memory usage at the time AP server started (the assumption of ~300MB is my experience of testing Web app deployed in JBoss 5.1GA, CentOS 5.3) |
| MinAPMem(1) = | APMemInit + MaxCU\*UMemSize/ (1\*1024\*1024) | 2,741.41 | MB | Minimum memory used at AP server in case of there's only 01 AP server |
| MinAPMem(2) = | APMemInit + MaxCU\*UMemSize/ (2\*1024\*1024) | 1,520.70 | MB | Minimum memory used at each AP server in case of there're 02 AP servers |
| MinAPMem(3) = | APMemInit + MaxCU\*UMemSize/ (3\*1024\*1024) | 1,113.80 | MB | Minimum memory used at each AP server in case of there're 03 AP servers |
| MinAPMem(N) = | APMemInit + MaxCU\*UMemSize/ (N\*1024\*1024) |  | MB | Minimum memory used at each AP server in case of there're N application servers |

***Hardware Profile Assumptions and Estimations***

**Heavy Knowledge Worker Profile**

A Heavy Knowledge Worker (HKW) is a very intense knowledge worker profile. Users who fit this profile have jobs that depend heavily on ProjectKit system/sub systems. For example: a QA will be considered as a HKW for *Project QCD* and *Report* modules.

|  |  |
| --- | --- |
| **Parameter** | **Assumption/Estimation** |
| DB server hardware | 4 processor, 2,600 MHz, 32 GB RAM |
| Application server hardware | 4 processor, 2,600 MHz, 4 GB RAM |
| Storage area network hardware | Hewlett Packard StorageWorks Enterprise Virtual Array  3 storage groups, 1 database per storage group, spread across 8 disk spindles using RAID0+1 |
| Peak processor usage | 75% |
| Peak disk usage | 45% |

**Medium Knowledge Worker Profile**

A Medium Knowledge Worker (MKW) is a normally intense knowledge worker profile. Users who fit this profile typically have medium number of daily tasks depends on ProjectKit system. For example: a QA will be considered as a MKW for *Scheduling* module.

|  |  |
| --- | --- |
| **Parameter** | **Assumption/Estimation** |
| DB server hardware | 4 processor, 2,600 MHz, 32 GB RAM |
| Application server hardware | 2 processor, 2,600 MHz, 4 GB RAM |
| Storage area network hardware | CLARiion FC4500  4 storage groups, 1 database per storage group, spread across 8 disk spindles using RAID0+1 |
| Peak processor usage | 75% |
| Peak disk usage | 45% |

**Light Knowledge Worker Profile**

A Light Knowledge Worker (LKW) is a light knowledge worker profile. Users who fit this profile typically have small number of daily tasks depends on ProjectKit system/sub systems. For example: a normal member will be considered as a LKW for *Report* module.

|  |  |
| --- | --- |
| **Parameter** | **Assumption/Estimation** |
| DB server hardware | 2 processor, 2,600 MHz, 4 GB RAM |
| Application server hardware | 2 processor, 2,600 MHz, 4 GB RAM |
| Storage area network hardware | CLARiion CX400  4 storage groups, 1 database per storage group, spread across 8 disk spindles using RAID5 |
| Peak processor usage | 75% |
| Peak disk usage | 45% |

### Server Sizing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Server set** | **Server** | **Hardware Item** | **Quantity** | **Unit** | **Specification** | **Note** |
| **Minimum** | Database Server | Server hardware | 1 | server | 2 processor, 2.6GHz, 2 GB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS | No replication, no RAID |
| Application Server | Server hardware | 1 | server | 2 processor, 2.6GHz, 4 GB RAM |  |
| Storage drive | 1 | server | 40GB, 100 IOPS |  |
| Httpd Server | Server hardware | 1 | server | 1 processor, 2.0GHz, 512 MB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |
| SSO Server | Server hardware | 1 | server | 1 processor, 2.0GHz, 512 MB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |
| **Recommended** | Database Server | Server hardware | 1 | server | 4 processor, 2.6GHz, 32 GB RAM |  |
| Storage drive | 8 | disk | 73GB 15K SAS drive | No replication, RAID1+0 |
| Application Server | Server hardware | 3 | server | 4 processor, 2.6GHz, 4 GB RAM |  |
| Storage drive | 3 | server | 40GB, 100 IOPS |  |
| Httpd Server | Server hardware | 1 | server | 2 processor, 2.0GHz, 2 GB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |
| SSO Server | Server hardware | 1 | server | 2 processor, 2.0GHz, 4 GB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |
| **Best** | Database Server | Server hardware | 2 | server | 4 processor, 2.6GHz, 32 GB RAM | One for active server, one for stand by |
| Storage drive | 8 | disk | 73GB 15K SAS drive | Using Replication, RAID1+0 |
| Application Server | Server hardware | 3 | server | 4 processor, 2.6GHz, 32 GB RAM |  |
| Storage drive | 3 | server | 40GB, 100 IOPS |  |
| Httpd Server | Server hardware | 1 | server | 2 processor, 2.6GHz, 4 GB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |
| SSO Server | Server hardware | 1 | server | 2 processor, 2.6GHz, 4 GB RAM |  |
| Storage drive | 1 | disk | 40GB, 100 IOPS |  |

## Multi-Tenant Support

Multi-tenant support allows ProjectKit system can be easily extended to SaaS platform. In this case, the future SaaS version of ProjectKit system will under the following pattern:

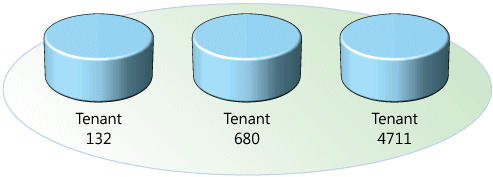
|  |  |
| --- | --- |
| **Category** | **Pattern** |
| SaaS approach | * Separate Databases |
| Security Patterns | * Trusted Database Connections * Secure Database Tables * Tenant Data Encryption |
| Extensibility Patterns | * Custom Columns |
| Scalability Patterns | * Single Tenant Scale out |

Table ‑ Patterns for SaaS Application

### SaaS Approach

***Separate Server, Separate Database***

The ProjectKit system uses approach of separate DB server, separate database to store tenant data.



Picture ‑Different database for each tenant or tenant group

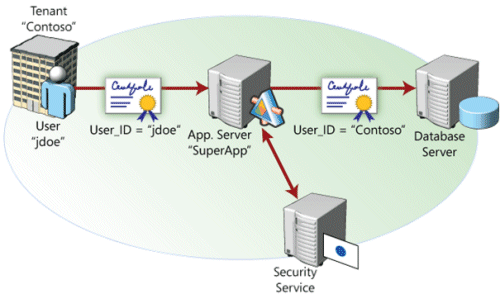
### Security

The security patterns rely on three underlying patterns to provide the right kinds of security in the right places:

* *Permissions*: Using access control lists (ACLs) to determine who can access data in the application and what they can do with it.
* *Filtering*: Using an intermediary layer between a tenant and a data source that acts like a sieve, making it appear to the tenant as though its data is the only data in the database.
* *Encryption*: Obscuring every tenant's critical data so that it will remain inaccessible to unauthorized parties even if they come into possession of it.

***Trusted Database Connections***

ProjectKit system’s tenant security uses both *impersonation* and *trusted subsystem account* access method, this approach involves creating a database access account for each tenant, and using ACLs to grant each of these tenant accounts access to the database objects the tenant is allowed to use. When an end user performs an action that directly or indirectly requires a call to a database, the application uses credentials associated with the tenant account, rather than credentials associated with the end user. The database server does not distinguish between requests originating from different end users associated with the same tenant, and grants all such requests access to the tenant's data. Within the application itself, security code prevents end users from receiving and modifying any data that they are not entitled to access.



Picture ‑An application connects to a database using a combination method

***Secure Database Tables***

To secure a database on the table level, use MySQL GRANT command to grant a tenant user account access to a table or other database object:

|  |
| --- |
| GRANT SELECT, UPDATE, INSERT, DELETE ON [TableName]TO[UserName] |

This adds the user account to the ACL for the table. The end users are associated with the security contexts of their respective tenants, so this only needs to be done once, during the tenant provisioning process; any end user accounts created by the tenant will be able to access the table.

***Tenant Data Encryption***

Tenant data is protected by *encrypting* it within the database, so that data will remain secure even if it falls into the wrong hands.

### Extensibility

ProjectKit system use separate DB schema for each tenant, thus, it is very easy to extend DB for new tenants with no system constraint.

### Scalability

***Single Tenant Scale out***

Tenant databases of project management may grow very large enough, so we may need to preserve an entire server with a single database that serves a single tenant.

## System Configuration

### System configuration file list

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Name** | **Path** | **Description** |
| **1** | web.xml | /WebContent/WEB-INF/ | Provides configuration and deployment information for the web components that comprise ProjectKit web application |
| **2** | … | … | … |

Table ‑ - System configuration files