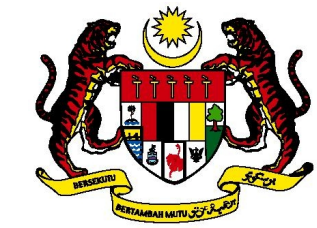

MAMPU

MALAYSIAN PUBLIC SECTOR OPEN SOURCE SOFTWARE (OSS) PROGRAM

WEB APPLICATION GUIDELINES

APRIL 2008

COPYRIGHT

Table of Contents

ABBREVIATIONS AND TERMS

Abbreviation	Full term	Description
3G	third generation	Third generation of mobile phone standards and technology
AGPL	Affero General Public License	A free software license published by the Free Software Foundation.
API	application programming interface	A source code interface that an operating system, library or service provides to support requests made by computer programs.
BSD	Berkeley Software Distribution	A permissive free software licences.
CORBA	Common Object Request Broker Architecture	A standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.
CPU	Central Processing Unit	Processor of a computer.
CSRF	Cross-site Request Forgery	A computer security vulnerability.
CSS	Cascading Style Sheets	A style sheet language used to describe the presentation of a document written in a markup language.
CSV	Comma-separated values	A file type that stores tabular data.
DCOM	Distributed Component Object Model	A technology for communication among software components distributed across networked computers.
DOM	Document Object Model	A standard object model for representing HTML or XML and related formats.
GPL	GNU General Public License	A widely used free and open software license, originally written by Richard Stallman for the GNU project.
GPRS	General Packet Radio Service	A packet oriented Mobile Data Service available to users of Global System for Mobile Communications (GSM) and IS-136 mobile phones.
HA	High availability	A system design protocol and associated implementation that ensures a certain absolute degree of operational continuity during a given measurement period.

Abbreviation	Full term	Description
HTML	Hypertext Markup Language	A predominant markup language for web pages which provides a means to describe the structure of text-based information in a document.
ICT	Information Communications Technology	A broad subject concerned with technology and other aspects of managing and processing information.
JDBC	Java Database Connectivity	An API for the Java programming language that defines how a client may access a database.
LGPL	GNU Lesser General Public License	A free and open software license published by the Free Software Foundation.
MAMPU	Malaysian Administration Modernisation and Management Planning Unit	A government agency belonged to the Prime Minister Department which handle the functions of Public Sector administrative modernisation and human resources planning.
MIT	Massachusetts Institute of Technology	A permissive free software licences originating at the Massachusetts Institute of Technology.
MVC	Model-view-controller	An architectural pattern used in software engineering to isolate business logic from user interface .
ODBC	Open Database Connectivity	A standard software API method for using database management systems.
OLE-DB	Object Linking and Embedding, Database	An API for accessing different types of data stored in a uniform manner.
OS	Operating system	A system software that manages computer resources and provides programmers and users with an interface used to access those resources.
OSCC	Open Source Competency Centre	Non-profit centre established by MAMPU as single point of reference to guide, facilitate, coordinate and monitor implementation of OSS in the Public Sector.
OSS	Open source software	A kind of software which source code is made available to the general public and permits users to freely use, change and improve it, and to redistribute it in modified or unmodified form.

Abbreviation	Full term	Description
PC	Personal computer	A computer whose original sales price, size, and capabilities make it useful for individuals, intended to be operated directly by an end user, with no intervening computer operator.
PEAR	PHP Extension and Application Repository	A framework and distribution system for reusable PHP components.
PHP	PHP: Hypertext Preprocessor	A computer scripting language, originally designed for producing dynamic web pages.
RPC	Remote procedure call	A technology that allows a computer program to cause a subroutine or procedure to execute in another address space without the programmer explicitly coding the details for this remote interaction.
SID	Session Identifier	A piece of data that is used in network communications (often over HTTP) to identify a session.
SOA	Service Oriented Architecture	A computer systems architectural style for creating and using business processes, packaged as services, throughout their life cycle.
SOAP	Simple Object Access Protocol	A protocol for exchanging XML-based messages over computer networks.
SQL	Structured Query Language	A database computer language designed for the retrieval and management of data in relational database management systems, database schema creation and modification, and database object access control management.
TCP/IP	Transmission Control Protocol / Internet Protocol	Protocols developed by the United States Department of Defense research project to connect a number different networks designed by different vendors into a network of networks (the Internet).
UML	Unified Modeling Language	A general-purpose modeling language that includes a graphical notation used to create an abstract model of a system.
URL	Uniform Resource Locator	A compact string of characters used to identify or name a resource typically in the network.

Abbreviation	Full term	Description
VPN	Virtual Private Network	A communications network tunneled through another network, and dedicated for a specific network.
W3C	The World Wide Web Consortium	The main international standards organization for the World Wide Web.
WAP	Wireless Application Protocol	An open international standard for applications that use wireless communication.
WCF	Windows Communication Foundation	A programming framework used to build applications that inter-communicate.
XHTML	Extensible Hypertext Markup Language	A markup language that has the same depth of expression as HTML, but also conforms to XML syntax.
XML	Extensible Markup Language	A general-purpose specification for creating custom markup languages.
XSS	Cross-site scripting	A computer security vulnerability.
ZPL	Zope Public License	A free software license published by Zope Corporation.

1. INTRODUCTION

A web application is a dynamic extension of web or application server. The ability to update and maintain web applications without distributing and installing software on any of the client computers is a key reason for their popularity and vast adoption in today's application development approach. As such, many applications nowadays are developed and implemented as web application.

A significant advantage of building web applications to support standard browser features is that they should perform as specified regardless of the operating system or OS version installed on the given client. The application can then be written once and deployed almost anywhere, rather than having to create separate clients for Linux, Unix, Mac OS, MS Windows, and other operating systems. In addition, web applications are typically storing both the program and data on the centralized hosting server, make it easy to maintain and backup, at the same time require very minimal disk space on the client PC.

1.1 Purpose Of This Document

These Web Application Guidelines were developed by Open Source Competency Centre (OSCC), MAMPU to assist Government agencies involved in the development and implementation of web based applications to base their decisions on the most current and best available practice to ensure that the web applications to be designed to meet the following:

- adhere to world class open standard
- meet users' expectations
- optimize user experience
- maximum usability

- optimum accessibility
- scalability
- interoperability
- security
- maintainability

1.2 Usage Of This Document

These guidelines are intended to be used as reference for related web application projects tender specifications, as well as for internal web applications development and maintenance projects in the Government agencies.

1.3 Scope And Applicability

These guidelines are intended to cover on-line hosted applications accessed with web browser or browser-equivalent clients.

The primary audiences for these guidelines are ICT directors, ICT managers, web project managers, system analysts, system designers, web application developers, web system administrators, and all related parties.

It is advisable to follow at least the 5-step implementation process for any web application project, namely:

1. Analyze the requirements
2. Determine application functionality
3. Adopt suitable web architecture

4. Design web application

5. Deploy web application

These guidelines are based on, and applicable to, the above 5-step implementation processes.

2. ANALYZE THE REQUIREMENTS

Before implementing any web application, it is important to estimate the web application's requirements and demands. The web applications should then be designed with attempts to provide the best performance and scalability to meet those requirements and demands. The resulting web application should be engaging, relevant and appropriate to the targeted users.

In the initial stage, the project team should use all available resources to better understand users' requirements. It is a golden rule that the greater the number of exchanges of information with potential users, the better the analysts and developers understanding of the users' requirements. The more information that can be exchanged between the project team and users, the higher the probability of having a successful web application. Source of information exchanges could include customer support lines, customer surveys and interviews, bulletin boards, sales people, user groups, trade show experiences, focus groups, etc.

The information gathered from exchanges with users can be used to build "use cases". Use cases describe the things that users want and need the web application to be able to do. The following provides an outline of a process for creating use cases:

1. Identify all the different users of the system .
2. Create a user profile for each category of user, including all the roles the users play that are relevant to the system.
3. For each role, identify all the significant goals the users have that the system will support.
4. Create a use case for each of the goals.
5. Structure the use cases.
6. Review and validate the use cases with users.

The Unified Modeling Language (UML) use case diagram can be used to graphically represent an overview of the use cases for a given system.

Listed in the table below are some of the open source diagramming tool that can be used to draw UML diagrams.

Software & License	Platform	Description
Dia (GPL licensed)	Linux, Windows, Mac OS	A general-purpose diagramming software which can be used to draw many different kinds of diagrams including entity-relationship models, UML diagrams, flowcharts, network diagrams, and simple circuits.
StarUML (GPL licensed)	Windows	Supports most of the diagram types specified in UML 2.0.
Use Case Maker (LGPL licensed)	Windows	Tool that helps to write organized use cases and to maintain related requirements.

Table 2.1 Open source tool recommended to draw UML diagram

2.1 Categorizing Functionality

Most users define "usability" as their perception of how consistent, efficient, productive, organized, easy to use, intuitive and straightforward it is to accomplish tasks within the system. The project team should ensure that the web application functionality meets user expectations, especially related to navigation, content and organization.

Most web applications provide services that fall into one or more of the following categories:

Information oriented – provides news, information and contents to the users. Usually does not need to track the identity of the user. Examples are portals, blogs, knowledge bases, directories, repositories, etc.

Service oriented – provides personalized service to the users. Keep track the identity of the user. Examples are forums, webmails, groupwares, helpdesk

systems, social networking systems, etc.

Transaction oriented – involves extensive interaction with the users which needs to follows process flows. Need to keep track the identity as well as navigation flow of the users. Example are workflow systems, online assessment systems, e-commerce systems, etc.

Different categories of web applications will have different looks and feels to fit for its purpose. Information oriented web applications emphasizes on its user interface, which need to be attractive and simple to use. On the other hand, service oriented web applications emphasizes on its personalized features and access speed. Transaction oriented web applications need to ensure the process flow is followed, and keep track on every user action in the system.

To avoid expensive reengineering of a web application, it is crucial when first designing the application to identify the features and functionalities it needs to provide.

Some common features in web application includes dynamic content generation, personalized display of information, accept user input, file uploads, session management, user authentication, database interaction, access control, integration with external services, etc.

2.2 Determining Usage And Traffic Capacity

It is important to identify the users of the web application, and to estimate and determine its usage and traffic capacity. With this information, we can plan for the suitable web architecture and platform to host the web application, including the choice of operating system, file system, web server application, database server, etc.

By determining the web application usage and traffic, the security level, downtime tolerance, maintenance schedule, backup schedule and necessity for load balancing

can then be properly planned.

If the traffic capacity is high, the system analyst should make reference to the most recent benchmarking report on operating system comparison, file system comparison, web server application comparison, and database server comparison, to look for the one which supports high traffic capacity with low system resource usage. This is to ensure system stability and accessibility during peak time of usage.

The information gathered should be used to justify the server architecture specifications to be implemented for the web application.

The OSS Reference Architecture published by OSCC, MAMPU provides good reference information and should be referred to during this stage of web application planning.

2.3 Planning For Scalability

In the case of web applications, scalability is the capacity to serve additional users or transactions without fundamentally altering the application's architecture or program design. If the application is scalable, steady performance can be maintained as the load increases simply by adding additional resources such as servers, processors or memory.

The web application should be planned and designed for scalability as one of the basic requirement. This is to overcome possible bottlenecks that might limit or affect the web application's performance.

2.4 Planning For Server Platform Independence

It is strongly recommended that the chosen platform for web application development must abide to open standards and free from vendor specific closed

standards. The programming language chosen should also be operating system platform independent.

This is to ensure that:

1. **The web application to be implemented should not be locked-in by the server platform vendor**. The web application should no way be disastrous on its maintenance and lifespan even if the particular vendor discontinues support to the platform used or makes significant changes to the platform that might not be compatible to previous versions and affect the usability of the web application.

2. **High level of portability can be achieved across server operating system**. The web application should be able to deploy in different operating systems (such as Linux, Unix, Windows, etc.) with the same piece of source codes.

Example of web application platforms that meet this requirement are PHP, Java, Python, Perl, Ruby on Rails, etc.

2.5 Planning For Browser Compatibility

To ensure its usability and accessibility, the web application should be well tested to work well with all major web browsers such as Firefox, Safari and Internet Explorer.

The final web pages generated by the web application to be sent by the web server to the web browsers should be valid to the specifications developed and published by the World Wide Web Consortium (W3C) . W3C provides free online “markup validation service”. There are also some standalone markup validation tools, and certain web application development tools that have built-in markup validation feature.

Validation Service	URL	Description
Markup Validation	http://validator.w3.org	Checks the markup (HTML, XHTML, …) of Web documents.
CSS Validation	http://jigsaw.w3.org/css-validator	Checks Cascading Style Sheets (CSS) and (X)HTML documents with style sheets.
Markup Validator	http://www.markupvalidator.com	Free front-end application for W3C Validation Services to simplify validation of single pages or entire websites, checking XHTML, HTML and CSS markup, as well as broken links.

Table 2.2 W3C Markup Validation Services

Beside that, the web pages generated by the web application should not use Document Object Model (DOM) method that is only supported by singular web browser. The DOM method used should equally work well and behave in the same manner for all major web browsers, including Firefox, Safari and Internet Explorer.

The web application should also avoid using plug-ins that are specific to singular operating system or singular web browser. All plug-ins used by the web application should be installable and usable in major operating systems (Linux, Mac OS, Windows) and major web browsers (Firefox, Safari, Internet Explorer).

2.6 Designing For User's Typical Connection Speed

The web application should be designed for the network connection speed of most users. This is because its usability and accessibility is very much depend on the user's connection speed. It could not be too slow for the user to access and navigate the web pages.

The system analyst should take into consideration if the web application is going to be accessed from:

- local area network (typically high connection speed)
- virtual private networks (depends on the VPN tunnel speed)
- broadband Internet (typically medium connection speed)
- 3G Internet (typically medium connection speed)
- 56kbps dial-up Internet (typically low connection speed)
- GPRS or WAP network (typically low connection speed)

2.7 Designing For Commonly Used Screen Resolutions

The web application should output its display for the most commonly used screen resolution of user's monitor. If the output screen is designed for high resolution and the actual screen resolution is low, the user will be unable to view the full screen, a lot of scrolling would involve and navigation will be difficult. On the other hand, designing for low resolution screen will need to compromise on the amount of information displayable on the screen at the same time.

At the time of writing, most users in Malaysia set their screen resolution to 1024x768 pixels or above, therefore it is advisable to design the web application for display on 1024x768 pixels screen.

3. DETERMINE APPLICATION FUNCTIONALITY

There is a broad range of operations or functionality a web application should provide. Some key elements that can affect the basic performance and scalability of a system are:

- Execution of data transactions
- Web page caching
- High-availability requirements
- Session integrity requirements
- Multilingual requirements

3.1 Analyzing Data Requirements

Once the use cases or scenarios are established in the requirement analysis phase, the system analyst should analyze the application's data requirements, such as read, write, update and delete operations, including any real-time transactions that must be supported.

When setting up data stores, several options are available for different types of data. The most common choices for storing data are as below:

Data store	Examples	Suitability
Relational databases	MySQL, PostgreSQL, Oracle, DB2, Informix, Sybase, MS SQL	Large amount of data that involves a lot of read and write actions.
Directory servers	OpenLDAP, OpenDS, Oracle Internet Directory, Tivoli	Data which needs significantly more read operations for relatively static information.
Flat files	SQLite, XML, CSV	Small amount of data. Can also be used to facilitate extract-transfer-load operation of data to another system.

Table 3.1 Options for data store

With an existing database, it is helpful to map out the necessary read/write dependencies and how specific data will be accessed by the web application. If a database contains a number of tables that parts of the application need to access, or it stores collections of unrelated types of data, partitioning the data into appropriate tables allows you to move them to discrete database servers, where they can be scaled individually when the data volume become huge.

Below are some of the commonly used database servers that support table partitioning:

Database server	Version
MySQL	5.1 and above
PostgreSQL	8.1 and above
Oracle	9i Enterprise and above
IBM DB2	9 and above
Informix	7 and above
Sybase	Adaptive Server Enterprise 15.0 and above
MS SQL	2000 and above

Table 3.2 Database servers that support table partitioning

3.2 Reducing Or Isolating Real-time Updates

Real-time update is an important functionality which enable changes made by one

user to be automatically and immediately updated to all the users using the same system at the same time. However, real-time updates is usually very costly, as they consume a lot of server computation and communication resources.

Therefore, once the database interactions are determined, non-critical functions should be identified and designed to be processed offline whenever possible to reduce or isolate the system's real-time update requirements.

3.3 Using Dynamic Content Accelerator

Web application which generates web pages on the fly exerts a major toll on server resources. This might result in bottlenecks at the server and slower response times for end users.

Dynamic content accelerators are used to address the latencies inherent in dynamic sites. These server-side accelerators intelligently feed data to application servers, letting pages be created much faster than they are on non-optimized sites by taking advantage of the fact that much of the content in a dynamically generated page is actually reusable and wise to cache for.

By implementing a suitable dynamic content accelerator, web loading speed, performance, stability and scalability can be tremendously increased.

Examples of dynamic content accelerators are Zend Optimizer, e-Accelerator for PHP, Akamai, etc.

3.4 Determining For High-Availability Requirement

High-Availability (HA) is crucial for mission critical web application which requires maximum reliability and zero downtime. HA should cover the application servers as well as the underlaying database servers, and robust to eliminate any possible single point of failure.

However, implementing web application that provides HA incurs additional cost on redundant components such as servers, disk storage and network devices. Maintenance to HA system is also comparatively more complicated. Requirement for HA should be studied carefully and only implemented when applicable. Nevertheless, the system should be designed in the way that can be easily adoptable to HA option, regardless HA would be implemented during the initial stage or in future.

3.5 Maintaining Session Integrity

We are all aware that the HTTP protocol is stateless, which means that the web application cannot correlate between browser access to different sections of a web application or web site. In essence, this means that users traversing a certain flow in an application cannot be identified between one stage and the next.

Most web application platforms support session mechanism as solution to preserve certain data across subsequent accesses of same user. Session is widely used to identify the user and register an arbitrary number of variables to be preserved across requests.

Complication arises when the web application is deployed on multiple servers, as in clustering, load-balancing and HA setup. Session integrity mechanism need to be in place to ensure that the user is authenticated in all the servers, and that any information captured during a user's session is made known to all the servers.

Session clustering is the technology which allows multiple web application server instances to share a common pool of sessions. The system analyst should carefully plan for session clustering to maintain session integrity if the web application were to deploy across multiple servers.

3.6 Considering Multilingual Support

Although most web applications may not have immediate needs for supporting multiple languages, it is a nice practice and increases value of the system to have this ability ready or built-in. Issues to consider when implementing multilingual support including:

- **Globalization support**. Globalization is an activity to make the application generally localizable to multiple locale. Things to consider are identifying control labels that are not universal, such as the link menu, or content of an article.

- **Localization support**. Localization is the activity of adapting an application to a locale. This is further broken into two categories – localization of interface and content.

- **Localization of interface**. Localization of the interface is an activity of making sure a resource is translated to another language with high fidelity. Examples of resources that require this are button labels, article text, article author names, etc.

- **Localization of content**. Localization of content is the ability to publish content that has high relevance to a particular locale but not in other locales. For example, a news site in Malaysia will publish news that may not interest anyone outside Malaysia. This however is not a multilingual issue, but consequential by making an application multilingual.

- **Translation fidelity**. As smart as the translators come, some content simply cannot be translated. The web application might need to identify area which are non-translatable, and provide option for user to view the resource in the original language.

- **Performance**. Providing support for multilingual normally incur additional

processing and result in performance degrade. The web application need to find out ways to minimize the impact on performance if multilingual support is required.

- **Scalability**. Scalability in multilingual support generally refers to the work required to support additional languages. This has implications to other requirements such as performance, maintainability and extensibility.

4. ADOPT SUITABLE WEB ARCHITECTURE

It is important to select and adopt suitable web architecture before designing the web application, because switching the web architecture after development will involve substantial re-engineering that will incur numerous cost, time and effort.

4.1 2-Tier Architecture

In the early days of web computing, most websites deployed a 2-tier architecture, which consisted of a web server that processed HTTP requests and a database server that provided a back-end data store. Application logic that served the website resided on the web server, which interacted directly with databases and generated dynamic web pages based on the query results.

Advantage of using a 2-tier model is that, it is relatively simple when designing and deploying a small system. However, web application developed with 2-tier architecture is very difficult to troubleshoot and maintain. Beside that, unacceptable response times can occur if a server cannot deliver simple page requests just because it is busy completing data updates or committing transactions. Because of its simplicity, 2-tier architecture is still vastly in use nowadays, but it is only recommended for simple application only.

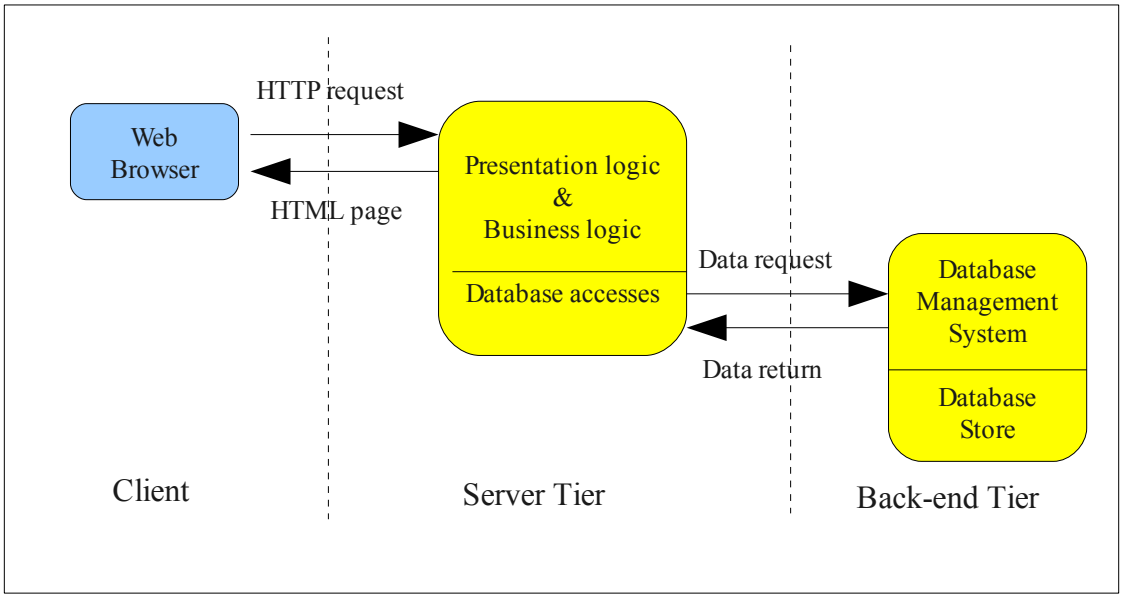

Figure 4.1 2-Tier Architecture

4.2 3-Tier Architecture

Separating the input/output-bound web operations (such as displaying static content) from processor-bound activities (such as performing user transactions) by deploying a 3-tier architecture effectively isolates and resolves scalability and maintainability bottlenecks in both of the 2-tier scenarios described above.

The 3-tier architecture model adds an application server tier to handle the business logic of a web application. With a 3-tier architecture, adding more web server tier machines can address the problem of slow static web page response times. If response times for processing transaction requests are slow, adding more application-server tier machines can improve their performance.

Another benefit of the 3-tier architecture is the ability to designate separate classes of machines for different tiers. The web tier machines can be simple single-processor or dual-processor machines, while the application-server tier machines can be mid-to-large enterprise-class servers with four or more processors each.

One potential downside of the 3-tier architecture is that it introduces additional

latency or processing delays, because processing requests and returning web pages to users require more communication between components. Most web requests, however, are already routed through several processing points on their way to the web server; adding one or two more will not create a noticeable difference to the end user. In addition, caching results or other static information accessed frequently by users can help alleviate latency, and adding a third tier provides additional ways of scaling.

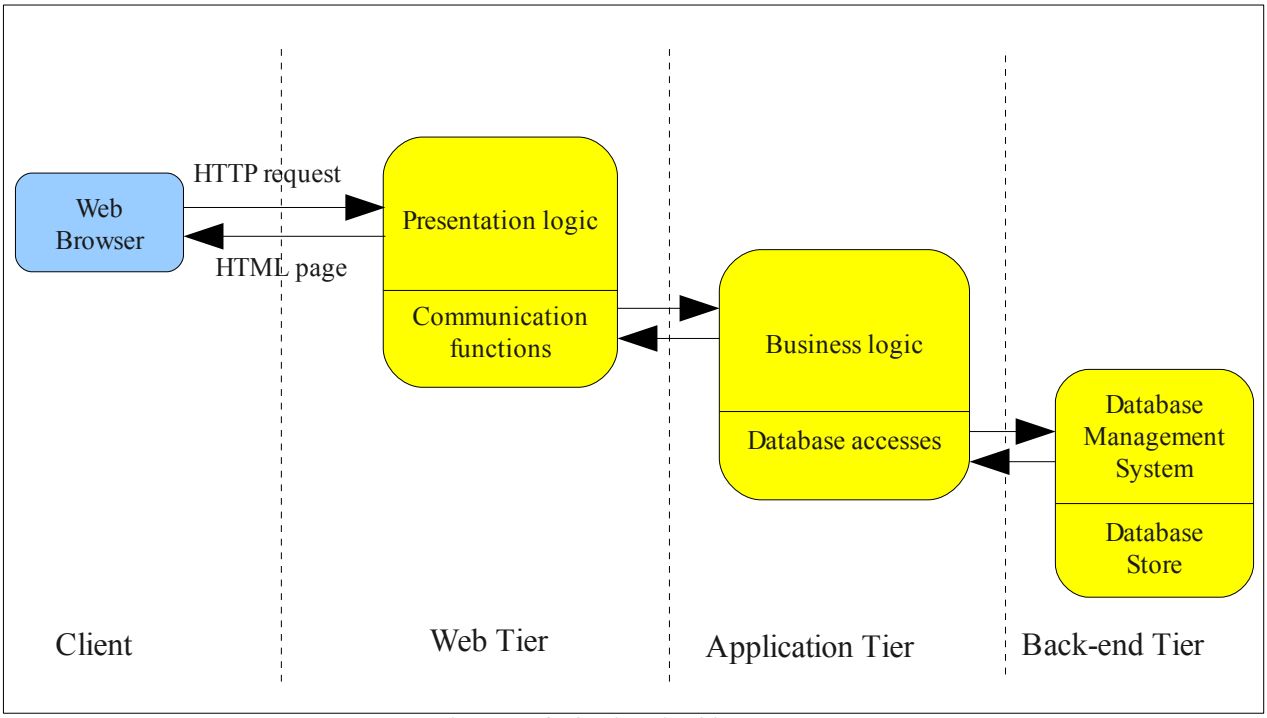

Figure 4.2 3-Tier Architecture

4.3 Model-View-Controller (MVC) Architecture

The main aim of MVC architecture is to separate the business logic and application data from the presentation data to the user. At first glance, the MVC architecture may seem similar to the 3-tier concept, but topologically they are

different. A fundamental rule in a 3-tier architecture is the client tier never communicates directly with the data tier; in a 3-tier model all communication must pass through the middleware tier. Conceptually, the 3-tier architecture is linear. However, the MVC architecture is triangular: the View sends updates to the Controller, the Controller updates the Model, and the View gets updated directly from the Model.

MVC is often seen in web applications, where the View is the actual HTML page, and the Controller is the code that gathers dynamic data and generates the content within the HTML. Finally, the Model is represented by the actual content, usually stored in a database or XML files. By decoupling Models and Views, MVC helps to reduce the complexity in architectural design, and to increase flexibility, reusability and maintainability.

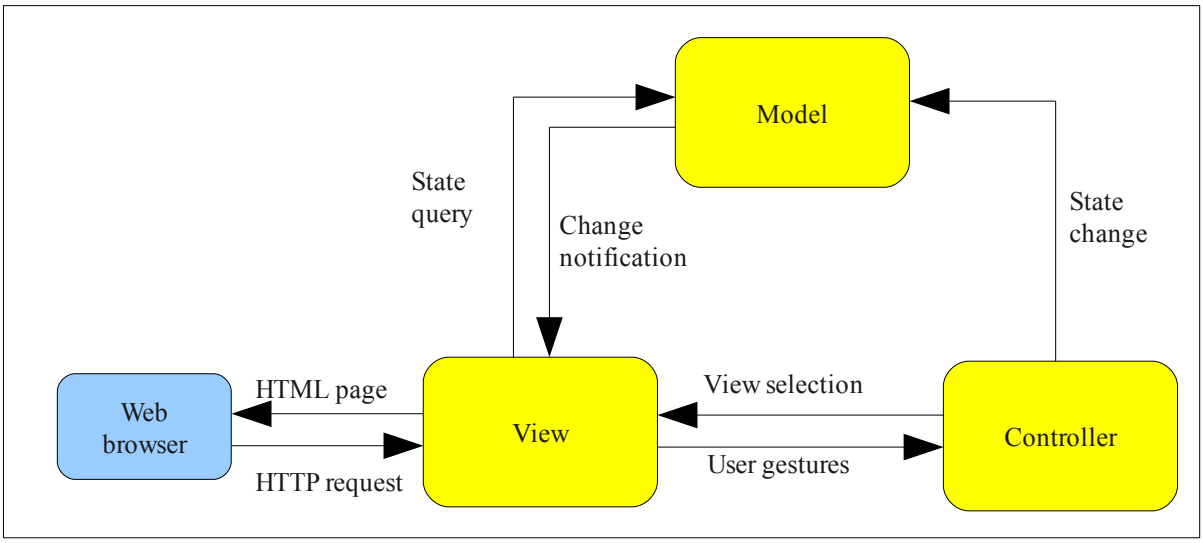

Figure 4.3 MVC Architecture

4.4 Service-Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is an architectural style for creating and using business processes, packaged as services, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications to exchange data and participate in business processes. These functions are loosely

coupled with the operating systems and programming languages underlying the applications. SOA separates functions into distinct units (services), which can be distributed over a network and can be combined and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services.

SOA may be implemented using a wide range of technologies, including SOAP, RPC, DCOM, CORBA, Web Services or WCF, and it can be implemented using one or more of these protocols.

Designing the web application to be SOA-ready or SOA-enabled is worth the consideration, as it will enable the web application to interoperate and integrate closely with other SOA-enabled applications, and allow it to reuse services and resources already available in other applications.

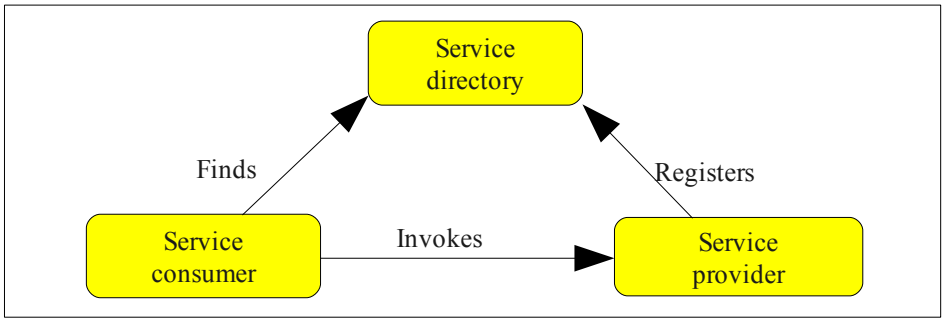

Figure 4.4 High-level SOA Concept

5. DESIGN WEB APPLICATION

Most web applications have roughly similar characteristics. They allow users to dynamically request and retrieve data, and provide a mechanism to store session information. At the heart of the web application is a data repository, usually a relational database. Services such as session state management, transaction monitoring, connection pooling, dynamic template processing, email and alerts are common elements in an application that typically requires storage or database components.

5.1 Designing For Scalability

Scaling a web application usually involves maximizing data throughput while maintaining data integrity. To maximize throughput, the system analyst can employ techniques like caching, queuing, connection pooling and connection pipelining. For example, connection pooling establishes a pool of open connections with the database, which can then be used and released rather than closed to reduce the overhead of constantly opening and closing connections. Pipelining similarly reduces connection overhead by opening a pool of connections between a web server and an application server, then using the connections to route the same type of information from different users. To maintain data integrity, fail-over techniques can be used together with best practice database design.

5.2 Using Modular Design

Designing an application to be modular rather than monolithic is crucial to facilitate scaling, especially horizontal scaling which requires being able to distribute load among similarly configured components, such as web or application servers. When independent processing phases are decoupled and allowed to feed each other via communication or messaging protocols (such as using XML), these

components can be separated onto different servers, as necessary, to improve performance and scalability. Evaluating the performance and scalability characteristics of individual vendor products used in the web, application, and database tiers of an application will also help determine how well they perform and scale either horizontally or vertically.

5.3 Adopting Suitable Framework For Development

Frameworks are extensible module sets supporting rapid development or repetitive requirements in a reliable way though reusable skeletal design pattern realizations. By adopting suitable framework, developers can concentrate on the problem domain rather than on low-level implementation details, while providing concrete benefits beyond the associated acquisition and learning curve costs.

Frameworks are comparable in terms of documentation quality, extensibility, modularity, maturity, and white/black box properties. The most useful frameworks supporting web application development can be categorized as web application and user-interface frameworks, persistence frameworks, and lightweight containers. Web application and user interface frameworks are the most directly relevant to dynamic content generation systems. Persistence frameworks such as Hibernate aim to allow programmers to efficiently retrieve and update database information through encapsulated objects rather than through direct SQL queries. Lightweight containers are frameworks that leverage inversion of control to simplify component-based development. For example, the open source PicoContainer and Spring frameworks are highly regarded lightweight container frameworks.

Below are some of the web application frameworks commonly used by developers nowadays. The license of the frameworks used might have governance to the license scheme and distribution obligation of the web applications. Reference of open source related licenses can be obtained from the OSS Implementation Guidelines published by OSCC, MAMPU.

Framework	Programming Language	License
Cocoon	Java	Apache
Struts	Java	Apache
Aranea MVC	Java	Apache
Google Web Toolkit	Java	Apache
Hamlets	Java	BSD
ItsNat	Java	AGPL 3
JBoss Seam	Java	LGPL
JZeno	Java	LGPL
OpenXava	Java	LGPL
PicoContainer	Java	BSD
Play!	Java	LGPL
RIFE	Java	Apache
Spring	Java	Apache
Stripes	Java	LGPL
Tapestry	Java	Apache
ThinWire	Java	GPL
Ajile	JavaScript	GPL
Helma	JavaScript	Apache
Widgetplus	JavaScript	GPL
Catalyst	Perl	GPL
Akelos	PHP	LGPL
CakePHP	PHP	MIT
Code Igniter	PHP	Apache
Drupal	PHP	GPL
EZ Components	PHP	BSD
Horde	PHP	GPL
Kohana	PHP	BSD
PRADO	PHP	BSD
Qcodo	PHP	MIT
SilverStripe / Sapphire	PHP	BSD
Symfony	PHP	MIT
Zend	PHP	BSD
Zoop	PHP	ZPL
CherryPy	Python	BSD
Django	Python	BSD

Framework	Programming Language	License
Pylons	Python	BSD
TurboGears	Python	LGPL
Web2py	Python	GPL
Zope	Python	ZPL
Camping	Ruby	MIT
Nitro	Ruby	BSD
Ruby on Rails	Ruby	MIT
ColdBox	ColdFusion	Apache
Fusebox	ColdFusion	Apache
Model-Glue	ColdFusion	Apache
OnTap	ColdFusion	BSD

Table 5.1 List of commonly used web application frameworks

5.4 Adopting Suitable Database Abstraction Layer

Database abstraction layer unifies the communication between the web application and databases by providing a single programming interface to the application developer and hiding the database specifics behind this interface as much as possible.

By adopting database abstraction layer, the web application will not be locked-in to specific database system, and changing the database behind the application would no longer require massive changes to the source code of the application to adopt to the new database system. This increases the portability, interoperability and scalability of the web application, since the same piece of application can work with multiple database systems.

Examples of database abstraction layer include OpenDBX, PEAR, ADOdb, JDBC, ODBC, OLE-DB, etc. Some web application frameworks provide database abstraction layer as one of their component.

5.5 Optimizing Connections

To achieve the best performance and scalability, the web application developers should minimize the total number of connections required to complete any requests or transactions in the web application. This applies to all the components from web servers processing HTTP requests and opening TCP/IP connections, to links from web servers to any pool of available application servers, to the server where database connections perform transactions and update data.

5.6 Making Sense Of Web Security

Web applications commonly face a unique set of vulnerabilities, due to their access by web browsers, integration with databases, and the high exposure of related web servers. Careful security measure ought to be taken during its development, especially to the following common issues:

5.6.1 Code Injection

Description: The web application user interface was exploited to the underlying operating system and results in the execution of arbitrary program code “injected” by the hacking user. The hacking user can then takes control and changes the program execution behaviour.

Prevention: Impose data validation on all user input. Escape illegal characters from the input. Encode data that is about to be output so that any potentially dangerous characters are made safe.

5.6.2 Remote Code-Inclusion

Description: Malicious codes hosted in remote site was made included into the current executing script. The hacking user can then takes control and changes the program execution behaviour.

Prevention: Disable global registration of variables. Disallow include parameter to have URL of unidentified remote site.

5.6.3 SQL Injection

Description: SQL query statement to the database was exploited by the hacking user to change program behaviour or to steal data from database.

Prevention: User input must not directly be embedded in SQL query statements. Instead, user input must be escaped, or parameterized statements must be used.

5.6.4 Cross-site Scripting (XSS)

Description: Data is entered by hacking user into the web application, which is later written back to another user. If the application has not taken care to validate the data correctly, it may simply echo the input back allowing the insertion of JavaScript codes into the other user screen.

Prevention: Impose data validation on all user input. Sanitize the user input by encoding all HTML special characters.

5.6.5 Cross-site Request Forgery (CSRF)

Description: The attack from hacking user works by including a link or script in a page that accesses a site to which the user is known (or is supposed) to have authenticated to forge the identity.

Prevention: To include a secret, user-specific token in forms that is already verified.

5.6.6 Directory Traversal

Description: Exploit through user-supplied input file names to access a computer file on the web server that is not intended to be accessible.

Prevention: Check, filter and sanitize the user input on file requests.

5.6.7 HTTP response splitting

Description: The web server was made to print a carriage return and line feed sequence followed by content supplied by the hacking user in the header section of its response. It can be used to perform cross-site scripting attacks and other exploits.

Prevention: To URL-encode all the strings before inclusion into HTTP headers.

5.6.8 Session Fixation

Description: Exploitation to the system to allow the hacking user to fixate or set to another person's session identifier (SID).

Prevention: Do not accept session identifiers from GET/POST variables. Store session identifier in HTTP cookie.

5.6.9 Session Hijacking

Description: Exploitation of a valid session key from another user to gain unauthorized access to information or services in the web application.

Prevention: Use a long random number or string as the session key. Regenerate the session id after user successful login.

5.6.10 Input Form Spamming

Description: Web form is being spam either manually or with automated program. If the web form input were used to sent out email to users, this will because a email spamming via the web application.

Prevention: Erase user input in web form after submission. Make use of captcha to prevent automated form submission by hacking user.

5.6.11 Executable Script And Program Upload

Description: The hacking user uploaded executable script and program onto the

web server and run the malicious program with URI request.

Prevention: Store all files uploaded by users outside the web folder in the server to prevent direct access to them from the web.

5.6.12 Defacement

Description: The web page was overwritten or replaced by the hacking user after gaining access to the server.

Prevention: Patching all known vulnerabilities of the web server and hardening its security.

Web application security scanner software can be used to analyze and diagnose for possible vulnerabilities in web application, such as those mentioned above. Some of the open source scanners include Grabber, Pantera, Paros, Spike Proxy, TestMaker, WebScarab, Wapiti and W3AF.

6. DEPLOY WEB APPLICATION

When it comes to deployment stage, the web application is ready to roll out and made available to the end users. Below are some recommended activities covering the scope of pre-deployment, during deployment and post-deployment of web applications.

6.1 Packaging For Easy Deployment And Distribution

The web application should be packaged for easy deployment and distribution. It should include documentation for step-by-step installation instruction in its distribution package. Installation or setup script should be provided to configure the initial application settings and setup the database of the application. The installation script should also check out the server for all the components required by the web application, and warn the user if any of the key components were not installed in the server yet. The system engineer should then install all the required components into the server prior to installing the web application, in order for the application to function properly.

6.2 Considerations For Shared Hosting

It is common for web application to share the same hosting server with other web applications. Compatibility check should be performed to ensure that all the web applications co-locating in the same server not working on conflicting components or rely on different versions of the same component.

Performance isolation need to be enforced so that one application would not be able to “steal” server resources from all other co-located applications, which include the CPU processing power, system memory, network bandwidth usage, etc.

Process isolation should also be enforced so that malfunction of a single web

application should not hang the whole server or affect the other running web applications.

6.3 Stress Testing The Web Application

It is wise to simulate the extreme usage capacity to the web application to determine its stability, robustness, availability and error handling capability under heavy load. Weakness found from stress testing ought to be resolved to ensure the service level to the web application is intact.

Example of open source web application stress tools include Pylot, JCrawler, Curl-Loader, The Grinder, Funkload, TestMaker, Siege, OpenSTA, etc.

6.4 Service Level Monitoring

Service level of web application is normally determined by its:

- uptime or service availability
- access response time
- server resource utilization level (CPU load, disk and memory usage, running processes, etc)

The web application service should be continuously checked for its service level and health, and the system administrator should be notified if any problem detected by the service level monitoring tool. The system administrator should act immediately to resolve any problem notified to maintain the service level to the web application.

MyNetWatch developed by OSCC, MAMPU is an example of open source service level monitoring tool that can be used for this purpose.

6.5 Conducting Web Analytics

Web analytics is the science of web application usage measurement and analysis, which deals with the gathering of web usage data, computation and presentation of matrics, and the exploitation of the results in order to improve the satisfaction of web application objectives.

By observing and analyzing the web application usage over a certain period of time, it is possible to extract users' behaviour patterns. Using these patterns as consideration for further development and enhancement to the application increases the chance to best meet the users' expectations and thus improve their overall satisfaction.

Examples of web analytics tools including AWStats, Webalizer, Analog, Google Analytics, WebTrends, etc.

Appendix: Checklist For Web Application Implementation

No.	Items	Description
1.	ANALYZE THE REQUIREMENTS	
1.1	Studies of "use cases"	
1.2	Functionality categories	
1.3	Scope of usage	
1.4	Estimated traffic capacity	
1.5	Scalability measures	
1.6	Server platform independence	
1.7	Browser compatibility	
1.8	Users' connection speed identified	
1.9	Users' screen resolution	
2.	DETERMINE APPLICATION FUNCTIONALITY	
2.1	Data requirements	
2.2	Real-time updates	
2.3	Dynamic Content Accelerator	
2.4	High-Availability requirement	
2.5	Session integrity maintenance	
2.6	Multilingual support	
3.	ADOPT SUITABLE WEB ARCHITECTURE	

No.	Items	Description
3.1	Web architecture	
3.2	SOA support	
4.	DESIGN WEB APPLICATION	
4.1	Scalability design	
4.2	Modular design	
4.3	Development frameworks	
4.4	Database Abstraction Layer	
4.5	Connections optimization measures	
4.6	Web security measures	
5.	DEPLOY WEB APPLICATION	
5.1	Packaging	
5.2	Shared hosting measures	
5.3	Stress testing result	
5.4	Service level monitoring tool	
5.5	Web analytics	