The meeting started with short proposal presentation of this technical report. After that, any attendances were free to gave their own opinions about the original prototype, expected characteristics of the new architecture, and prioritize all requirements.

The purpose of this requirements meeting is to get technical feedbacks and suggestions from the users of the original prototype and use those information as criteria for the 2nd frameworks selection and comparison. The selected frameworks will be analyzed in detail, implemented as prototype of the existing system, and evaluated in the following chapters.

2.5. System Requirement and Analysis

2.5.1. 1st priority (1)

- Avoid complex and hard to understand configuration files.
- Many small files are more preferable than less amount of large files.
- AJAX is optional but the system should be able to work without it.
- The framework's community should be active and the project must be currently running.

Most of the attendances chose to avoid basic pitfalls which, increases the complexity of the system as the first priority. One factor that affect the complexity is large files in the system especially, configuration files such as xml files. There are several ways to configure the framework but mainly, by xml file or annotation. Small amount, but large artifacts are harder to maintain than higher amount, but small and simple artifacts since, there are very few tools/IDE, which support xml auto-completion and verification. Unlike xml, configuration by annotation eliminates additional artifacts and easier to understand. However, the advantage of xml is the ability to manages the properties/parameters of deployed system without recompilation or redeployment. According to the feedback from user, convention over configuration framework is preferred, and possibly, zero configuration will be the best.

Asynchronous Javascript And XML (AJAX) is a web development method used on the client-side to create interactive web application. With AJAX, web application can send and retrieve the data from web server asynchronously (in the background) without any sign of page refresh. For the existing system, any frameworks have AJAX support are advantageous, but the system must still working perfectly, even without AJAX. This requirement was the old requirement brought from the original prototype.

Community and support also important. The chosen framework's community

should be active and provides support. For example,

- The official website should provide good introduction to the framework and resources for starters.
- The framework should release update version frequently and the project must be currently running.
- The community such as web board, developer forum, or mailing list should be active. The skillful members and framework development team should enthusiastic to support and provide the answers the starter. The developers of the framework should also look after bug report.

The reason behind given the first priority on complexity of the system was all attendances agreed that the new prototype should emphasize on convention over configuration, easy to maintain, and good supports from community.

2.5.2. 2nd priority (2)

- Component-based, inheritance supported architecture. All components must be reusable and should have inheritance structure.
- Delegation and navigation support.

Reusability is always first factor to indicates the quality of the software development and codes. For the backend and business logic, one way to obtains the reusability is through inheritance structure. Most of the web application view created by markup language such as HTML, XHTML, and JSP. All of those markups does not support inheritance structure, or any mechanism that provides the user interface reusability. One way to accomplished the 2nd priority requirement is component-based architecture framework.

In component-based aspect, each page considered as a component, and each page can be constructed with multiple components, which means, each component can be reused to construct pages, which shared some similarity as many time as needed without code duplication.

For example [Figure2.9], a basic login component composed of text boxes to receive input data from user (i.e. username and password), and a form submission button. Assume that there are two types of login page, which are login page for normal user, and login page for system administrator. These two types of login page have generalization relationship with basic login page, which means they share similarities. If the framework supports inheritance structure for user interface, these two pages can extend basic login page as a parent instead of duplicate the whole codes from the basic login page. The official name for the markup inheritance is template inheritance.

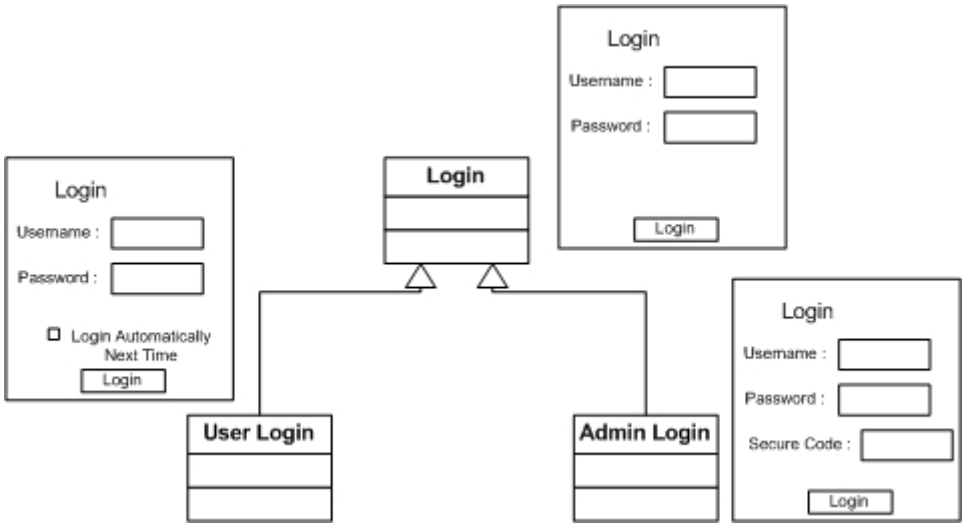

Figure 2.10.: Example of component-based, inheritance structure

Another way to fulfill this requirement is the framework should provides Portlet behavior User Interface or at least, provides Portlet API. Based on Portlet Specification [AH03], Portlets are web components-like Servlet, specifically designed to be aggregated in the context of composite page. Combining with CMS likes Portal, many Portlets are invoked in the single request of a Portal page. Each Portlet procedures a fragment of markup of other Portlets, all within the Portal page markup.

Inheritance supported architecture also provides several benefits for the system such as:

- Provides high reusability. Less code duplication.
- Provides high understandability and maintainability.
- Provides simpler page construction.

Another sub-requirement for the 2nd priority requirement is the framework should be able to manage the request delegation automatically. This request delegation is responsible for moving a particular request through the right class. With the delegation supports, the flow of the entire system is under control. For example, in the original prototype, the class ActionHandler.java is responsible for the request delegation. It will redirect user to the responsible action class. However, the example of the existing system is not the solution. The Action Handler is a normal Servlet, which handles the flow manually and roughly. The framework, which support better flow control is needed.

The main reason, why attendances gave the component-based architecture as the second top priority was to avoid the boiler-plate code or repetitive code which appeared in many places of the system. Component-based architecture also

provides great reusability for the user interface components. Business Process Management or Navigation Rules should be integrated to the existing system to handle the request delegation.

2.5.3. 3rd priority (3)

- Good IDE support such as auto-completion, UI builder, multiple project support, and etc.

Good IDE and tools support is another factor affects learning curve, productivity, errors, and maintainability of the project. Also, supported IDE and tools make presentation layer development more convenient. These are several examples of support features and their benefits:

- Auto-completion can be very productive and reduce human-errors.
- User interface builder provides easier way to create and manipulate user interface components such as drag & drop, readymade user interface component libraries, and more. These features lower the learning curve, and also provides high productivity. Some IDE provides simpler interface for configuration file modification, which reduces difficulty in configuration file handling.
- Code generator provides very productive way to generate invariant codes. For example, generates events bind to user interface components, Beans, CRUD application, or Test cases.
- Multiple project support.

2.5.4. 4th priority (4)

- Easy to test such as IDE support for test case generation, clear separation between presentation and navigation logic, and etc.

Every software development project includes testing phase after the implementation phase. This became the reason why attendances prioritized the requirement related to testing as fourth priority requirement.

The architecture of the prototype after integrates with the framework should be easy to test. The code of the presentation layer should clearly separated with the business logic or else, Unit testing and Integration testing will be difficult to distinguished. For example, MVC architecture provides clear separation between models, views, and controllers so the different level of testing can be distinguished easily.

The frameworks with IDE and tools supported for testing such as test case generation, or provides powerful testing API are advantageous.

2.5.5. 5th priority (5)

- Model-based generation support

Since there is a project currently running about generation of web-based prototypes for business applications, If the structure of the prototype after integrates to framework has static pattern and possible to generate using code generator theory, would be advantageous.

2.6. Summary

In previous sections of this technical report, seven famous frameworks are introduced. The figure below [Figure 2.10] shows the comparison of each framework in context of the requirements.

Frameworks/ Requirements	1	2	3	4
Spring MVC	⊘	⊘	⊘	✓
JSF	✓	✓	✓	⊘
Wicket	✓	✓	⊘	✓
Seam	✓	✓	✓	✓
Struts2	✓	⊘	✓	✓
Tapestry	✓	✓	✓	⊘
Stripes	✓	⊘	⊘	✓
JSP/Servlet	✓	⊘	⊘	✓

Figure 2.11.: Frameworks comparison against requirements

For the common pitfalls (1st priority requirement), almost of the frameworks passed, except Spring MVC and Stripes. Spring MVC has serious problem with very high configuration complexity (pure XML, configuration over convention), and Stripes has a small community and not actively developed.

For the component-based, inheritance structure architecture, and the request delegation supported (2nd requirement), all of the frameworks have Portlet API, so the Inheritance supported structure is possible. However, four frameworks have component-based architecture, which are JSF, Seam, Wicket, and