**Architecture constraints**

Development environment: Linux

Development platform: Java-EE

Architectural frameworks: Java Server Faces (JSF)

Version control management: Git

Development technologies:

* Java Persistence API (JPA)
* Java Persistence query language (JPQL)
* Hypertext Mark-up Language (HTML)
* Asynchronous JavaScript and XML (AJAX)

Architectural constraints

System architecture

Java EE reference architecture used to archive scalability, security and reliability

Frame work

The system will be developed under Java EE frame works which includes JSF, JPA and jPQL with addition of the Ajax frame work for [dynamic web pages](http://en.wikipedia.org/wiki/Dynamic_web_page) on the [client side](http://en.wikipedia.org/wiki/Client_side).

Time Constraint

Given 2 weeks for the functional requirements, 1 week for the architectural requirements, 4 weeks for the implementation of the system and 1 week for testing.

Web services

REST access channel will be used as is more lightweight and doesn’t require a lot of bandwidth, this enhances the goal of one of the core quality requirements which is accessibility to many students, thus they can access via phone device web browsers with easy.

Environmental Constraints

The system will be deployed at the University of Pretoria computer sciences server as an improvement to the current discussion board system.

Authentications

LDAP will be used for login authentication as the system is constraint to the University of Pretoria Students and already the Department have been using LDAP.

Database Technology

MySQL will be used as it enhances scalability , is open source ,thus will not cost university that much and have features for security that includes serialization, encrypting of passwords ,hashing and many more which can be implemented for strengthen security to the system.

Unit Testing

Maven (software project management) will be used for the system unit testing.

Technologies

Apache Maven

Blackboard pattern

To archive scalability, blackboard multiple processes to work closer together on separate threads, introduction of this pattern will help out multiple process of the buzz system to run efficiently as the pattern emphasizes multiple process working together

Layered Architecture

System will be separated through layers; there will be human interaction layer, services layer and process layer. Human interaction layer will handle interaction like receiving input from users, the service layer will provide the human layer with services like opening a buzz space and commenting on the buzz thread and lastly process layer will process services rendered for authorization and quality check like plagiarism. Separation through layers will enhance performance, manageability and reusability.

*Client/Server*

**For communication of the server which is buzz system with users, this pattern have benefits of security as all data will be stored on the buzz system server and ease of maintenance as server is responsible of repair with client knowing of damage.**