COS 301

Architectural Requirements for Buzz System

Group 4B: Phase 2

Authors:
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout
u11013878: Jaco Bezuidenhout

Supervisor:
Ms. Stacey Omeleze

`https://github.com/RenaldoV/COS301_Group4_B`
March 5, 2015

Contents

Chapter 1

Access and integration requirements

1.1 Access Channels

1.1.1 Human access channels

Thick client

-All resources need to be downloaded from the server.

Responsive design

-User Interface can be scaled according to device/size of the window to ensure readability and useability. E.g. Bootstrap.

Software needed

-Any modern browser that supports HTML & JavaScript E.g. Google Chrome, Mozilla Firefox.

1.1.2 System access channels

Other systems will access the system via the API provided as this system will act as a plugin that can be used on any other system that wishes to add a discussion board. RESTful will be the main API

Chapter 2

Architectural responsibilities

2.1 Architectural Responsibilities

2.1.1 Persistence Infrastructure

- As per the client requirements, some of the textual data generated or used in the buzz system will be maintained through persistent storage like a relational database. All media files and documents will then be stored on a separate directory on the server.
- The buzz system will also make use of LDAP (CS databases). So the CS databases have to share resources (cross-origin resource sharing) with the buzz system.

2.1.2 Reporting Infrastructure

- Send daily email notification to contributors of a thread
- Send instant email notifications to
  - Owner of a thread or post when a thread/post is
    - Updated
    - Deleted
    - Received a reply
    - Referenced

Chapter 3

Quality requirements

3.1 Quality Requirements

3.1.1 Critical

Scalability

The system is expected to have a maximum of 2000 users as specified by the client. Buzz Space is meant to encourage discussions on multiple topics at the same time by a large amount of users so the system needs to be able to handle many concurrent users too.

Monitor-ability and Audit-ability

The system should be monitor-able and audit-able, to enable lecturers to monitor the discussion boards and to see who posted what on the Buzz Space. Lecturers should be able to evaluate and monitor each user to be able to give the user a participation mark for the discussion board. Lecturers should be able to prevent irrelevant posts and discussions and should be able to see who were part of the discussion and notify them that it has been removed and should then be able to give them a warning.

This can all be achieved by ensuring that the system is monitor-able and audit-able.

3.1.2 Important

Reliability and Availability

Reliability and Availability is important in our system, as maintainability will then be less needed in an ideally reliable system. One of the key features of a discussion board is that it should be available 24/7. Users want answers and help as soon as possible, and therefore the system should be reliable and available at all times. If the system is very reliable and available, discussions that are of a time essence can be seen to as soon as possible. Reliability can be defined as the probability that a system will produce correct outputs up to some given time. A reliable system does not silently continue and deliver results that include uncorrected corrupted data. Availability means the probability that a system is operational at a given time. A highly available system would disable the malfunctioning portion and continue operating at a reduced capacity.

If the system is very reliable and available, users will be able to use the system with ease and the users will trust the Buzz system.

Security

Security is important because the evaluation of users' participation is involved. Not all of the users should be able to access these sensitive information and therefore certain users will be authorized to do certain actions, where other users will not be allowed to. Only users who are registered to the Buzz Space gave access and will be able to partake in the discussion boards. If the system is very secure, it will help to ensure Reliability and Availability.

Usability

The system must be as usable as possible so that users are encouraged to use it as much as possible. The extent to which the system can be made usable is affected by the priority of scalability and audit-ability because if the system is made scalable so that it can be used by all students at the same time, it is impossible to cater for 2000 student's needs and make it 100% usable for the mass. Usability is marked important because the participation of students can be greatly affected if the system is unusable and students will then lose marks.

Integrability

One of the main functional requirements of the buzz system is plug-ability, this goes hand in hand with the quality requirement integrability thus marked as important. The system need to be able to plug into any other systems and also need to use other systems where it can.

3.1.3 Nice to have

Testability

Testability cannot be fully implemented or given a high priority because quality requirements, scalability and auditability have been given high priority and that affects the extent to which the system can be tested. Users will test the system and give their feedback.

Performance

Performance is expected to be reduced as scalability takes priority. When the system is being used by a large amount of people at one time it will be acceptable for it to take a few seconds to do anything. For small amounts of concurrent users the system should still be fast.

Maintainability

Once the system is implemented it is not expected to need regular maintenance as reliability is an important quality requirement.

Chapter 4

Architecture constraints

4.1 Architecture Constraints

4.1.1 EE

The client has requested that we use the Java EE environment if possible when developing a large portion of the Buzz System.

Systems developed using Java EE is known to be fairly resource intensive, which may result in performance issues. The effect of this constraint may be amplified due to the time (un-optimised code) and system resources constraints (low hardware performance). A large portion of the students who are going to be developing the system to do have experience with using Java EE and learning to use this environment will take up implementation time from the already tight development schedule.

4.1.2 Time (Project deadlines)

Deadlines are set for the delivery of the project for which different phases of the project should be demonstrated and evaluated by the client.

Time is a major constraint, because the deadlines that were given are non-negotiable. The development of the system needs to be completed in the allocated time or else the developers would have failed the assignment given to them by the client. This will inhibit the quality of the implementation as the programmers will have to follow a strict schedule and will focus largely on the throughput of their work, instead of ensuring that they write good, maintainable and reliable code. If the development team run into an unexpected delay one, or many, of the quality requirements will have to be compromised to remain within the time constraint bound.

4.1.3 System Resources

The University will most likely not spend a large amount of money on a powerful server for the system.

The hardware on which the system will be running places a bound on the performance the developers can expect from the final product. This has to be taken into consideration when deciding which quality requirements are realistic.