Architecture

Release Information

Project: Engrade Quizzes
Internal Release Number: 1.0
Related Documents: Software Requirements Specification
Design > Security Worksheet
Glossary

Overview

What are the most important facts that a developer should know about this system architecture?

The system consists of three sub-systems:

- User Interface – provides user friendly interface for stakeholders using the system.
- User Account Management – Teacher/student account manager
- Quiz Management - Manages quizzes added by teachers

What software architecture style is being used?

2-tier web application: web server/app-server, database.

What are the ranked goals of this architecture?

1. Ease of integration
2. Extensibility
3. Capacity matching

Components

What are the components of this system?

The components of this system are clearly defined in this UML Model with Component Diagram.

The components of this system are listed below by type:

- Presentation/UI Components
- Application Logic Components
- Data Storage Components

Deployment

How will the components be deployed to processes and machines?

The deployment of components to processes and machines is clearly defined below:

- All-in-one server
  - Tomcat process
  - Database process

The deployment of components to processes and machines is clearly defined below:

- Load-balanced front-end servers
- Back-end server
  - Database process

What aspects/resources of their environment are shared?

2-tier same web and database physical machine server:

> Everything is on one server so all machine resources are shared by all components.
>
> All machines share the same bandwidth to the Internet. All machines access the same file server. So, if one component uses the resources heavily, other components may have to wait.

2-tier different web and database physical machine server:

> The web and database server are allocated with different and independent resources. Components may use either the web or database resource according to function used (e.g., page rendering, database commit, etc.).
>
> Machines may or may not share the same bandwidth. The two physical machines may have different internet service providers.

How are requests allocated to redundant or load-balanced servers?

We are not doing any load-balancing or redundancy for fail-over.

Load-balancing among front-end servers is handled by the web server that we can make very few assumptions about. However, once a user session is established, the same front-end server will be used for all requests during that session.

What alternative deployment configurations are possible?

The database could be moved to a different machine with a fairly simple change to a configuration file. Otherwise, nothing can be changed about the deployment.

Integration

How will components be integrated? Specifically, how will they communicate?

All of our code uses direct procedure calls or standard Python events. The database is accessed through a driver.

What architectural mechanisms are being used to ease future extensions or modifications?

We could change the database by switching drivers. Otherwise, extensions and modifications can only be done at the design level.

Architectural Scenarios

The following sequence diagrams give step-by-step descriptions of how components communicate during some important usage scenarios:

- User registration
- Quiz Management
- Site Configuration

Architecture Checklist

Ease of integration: Have mechanisms been provided for all needed types of integration?

Yes. In this system, all of the new components are designed to work together. And, the reused components are integrated via fairly simple interfaces.

Extensibility: What types of components can be added later and how?

See above.

Capacity matching: How has this architecture matched component resource needs to machines?

The database can be on a machine with RAID disks and a hot-swappable power supply, while the web front-end components can be on cheaper machines that could fail individually without causing system downtime. The front-end web servers and application server are not CPU-intensive, so they are not deployed to different CPUs. The database is disk-intensive, so it can be deployed to the same machine as the CPU-intensive application server, with only moderate competition for resources.

Has the architecture been communicated to the development team and other stakeholders?

No, the architectural design is still subject for a technical review. Feedbacks are welcome.