Web Application Development

Challenges of Web Application Development

- Web application development, no matter Internet-based or intranet- and extranet-based, presents unique challenges for developers.
- The major challenges include usability design, content-rich maintenance, security, integration with legacy systems, and fast application deployment. For Internet-based applications, there are additional two challenges— scalability and load balancing.

Challenges of WEB Development

- **Usability Design:**

  The usability of an e-commerce Web site, to a large extent, will determine the success or failure of the organization's Web presence.

  Ecommerce applications are designed for unknown users, unknown hardware platforms, and unknown software configurations at the users-side.

- **Content-Rich:**

  Most Web applications are content-rich.

  Content-rich applications require frequent updates and maintenance. A less frequently updated Web site quickly cast doubt on its visitors' mind about its accuracy and usefulness.

- **Scalability:**

  An Internet application runs in a different operating environment than a non-Internet based application does. For example different type of users will use the web application on different browsers.

- **Load Balancing:**

  In a multi-server Internet application, unbalanced workload on servers reduces system performance, reliability, and availability. Balancing system's load requires careful selection from an array of tools and techniques. There is no single silver bullet that can be applied to all application systems. Some of the load balancing techniques includes application partitioning and service replication.

Security:

Security is a major concern for Internet applications because of the open operating environment.

The following security issues must be addressed:

- **Privacy:** How to ensure that confidential data are safeguarded.
- **Integrity:** How to ensure that data consistency and accuracy are maintained when they are traveling on the network.
- **Authentication**: How to verify the true identity of the parties involved in a business transaction.

- **Access Control:** How to allow authorized users to access only the information they are allowed to access. How to prevent unauthorized access.
- **Non-Denial**: How to prevent denial of transaction submissions, either from the sending or receiving ends of the communication process.
- **Integrating Legacy Systems:**

  More and more organizations are linking their legacy systems, which may run on different computing platforms, to their Web applications.
- **Fast Development:**

  A well-designed quality Web application can be a competitive advantage.

Architecture Decision

- After a careful analysis of the system requirements and use cases, the decision on system architecture must be made.

- This decision must be made based on both the current needs and future development.

- There are many ways to layout a Web application architecturally. But there are three major models: (1) thin client, (2) fat client, and (3) distributed and component-based.

Architecture Decision

Thin Client: Client has minimal computing power. All of the business logic and rules are processed at the server.

- The client is a standard Web browser.
- This model is mostly used for Internet-based and some extranet-based applications because control over the client's configuration is lacking.
- The model gives developer greater freedom in system deployment and maintenance.

Fat Client:

- In this model, a fair amount of business logic and rules are executed on the client machine.
- Fat clients typically use dynamic HTML, Java applets, or .NET Framework components.
- Fat clients are used for some intranet applications that must provide customized services to certain user group.

Distributed and Component-Based:

Distributed and component-based architectures are used to support distributed object-oriented systems and Web services.

In previous two models (thin and fat clients) a business system is deployed at one location.

- The business logic for the application is implemented in a tightly coupled proprietary system.
- A distributed object system or a Web service, however, allows parts of the system to be located on separate computers, possibly in many different locations.