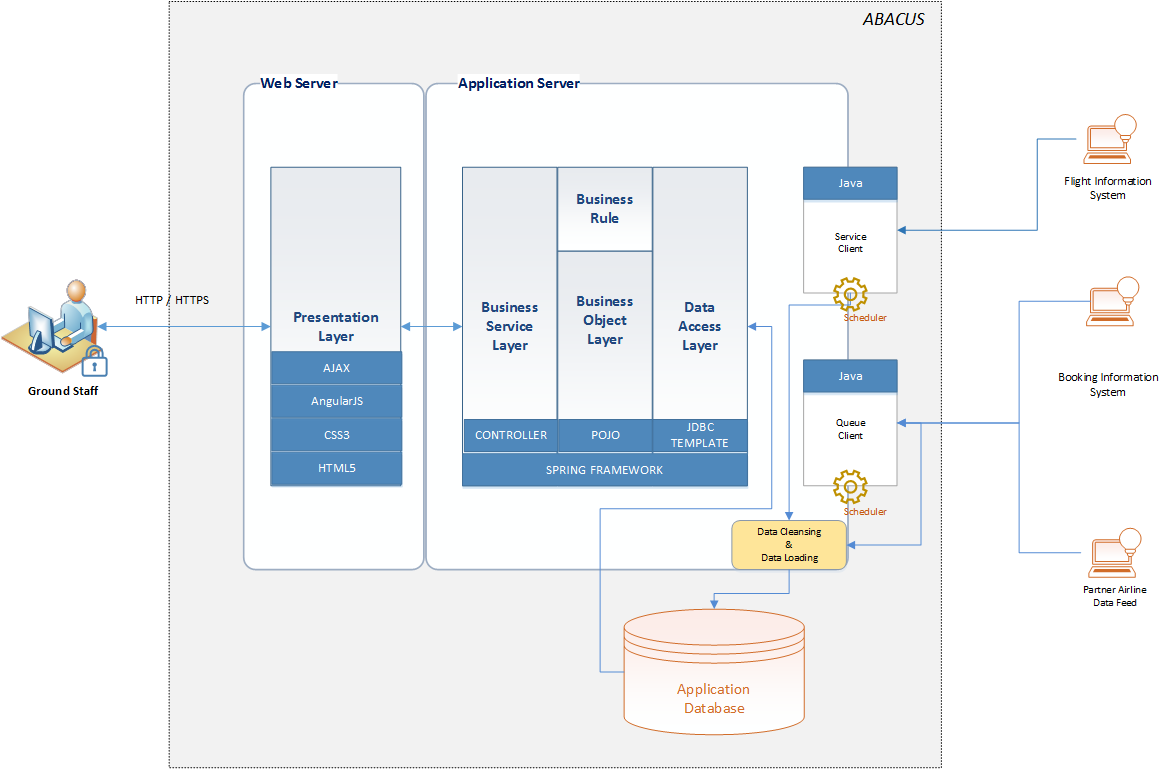
### Technical Solution Overview

In arriving at our solution, we have been guided by the following principles:

* Best-of-breed product components over components from the same stack to maximize functionality fitment
* Layered architecture: Each layer accessible only through well-defined interfaces, ensuring clear separation of logic and responsibilities
* Component Based Design: This is to ensure re-usability of functionality and to reduce redundancy thus also to improve maintainability.
* Emphasis on scalability, ease of use, process efficiency, flexibility, security
* Integration of various components using out of the box APIs and custom services
* An implementation approach that minimizes risk and optimizes the deployment of resources and switchover of the sites
* Testing as an Independent Verification and Validation process to ensure optimal quality



### Solution Components

**Presentation Layer**

* User Interface (Web / Browser) will be developed following HTML5 standards and CSS3. User Interface will be browser independent and will support all modern browser versions.
* Technology like HTML5 and CSS3 will allow designer / developer to make User Interface responsive. [This responsive feature will be implemented depends on business requirement and will be decided in design time]
* Industry standard Client side MVC based JavaScript framework (i.e. AngularJS) will be used to implement client side dynamic behaviour in very structured manner.
* AJAX (Asynchronous JavaScript and XML) will be used to transfer data between Presentation layer and Business Service Layer. This will ensure less network traffic during UI display and faster response time

*Technology Benefit*:

* *HTML5 and CSS3 will allow creating structured, readable and semantically accurate code. Application Maintenance will be much easier.*
* *HTML5 and CSS3 help designer to develop web pages compatible with different browser versions*
* *AngularJS will allow developing client side code following MVC design pattern, using two way data binding and templating.*

**Business Service Layer**

* Business Service Layer will be developed using Java. Java based Spring framework will be used as underling framework to support DI (Dependency Injection).
* Spring Controller will be used to develop client facing services and will expose service output in form of JSON object.
* Business Services will use business objects to execute specific business functionality.

*Technology Benefit*:

* *Spring allows to organize service tier objects*
* *No Special container is required*
* *Service can be developed and exposed using annotation only*
* *Service can expose JSON or XML objects as required through configuration only*
* *No external utility is required for Service development*

**Business Object Layer**

* Business Objects will be developed as simple Java Objects (POJO) and will be managed by Spring container. Internally Business Object will use business rule objects to execute different business functionality. Separate Business Rules will allow externalizing core business functionality.
* Business Rule Engine (BRE) product has not considered for this solution. Incorporation of new rule will be developer’s responsibility. External Rule configuration (through property files) will make development and integration smooth and easy.
* Business objects will use Data Access Layer components to communicate with Database.

*Technology Benefit*:

* *Spring allows to organize Business tier objects*
* *No Special container is required*
* *Business rule can be separated out from other layers*
* *Application enhancement and Maintenance will be easier*

**Data Access Layer**

* Data Access Layer will be used to access data from underlying database. Data Access Layer java components will be managed by Spring Container. And Spring JDBC template will be used to access data from database using database objects [Stored Procedure] or PL/SQL.

*Technology Benefit*:

* *Spring allows to organize Data Access tier objects*
* *No Special container is required*
* *JDBC Template allows to communicate with Database seamlessly*
* *Lesser amount of code ensure faster development cycle*

**Service Client**

* Service Client will be developed as simple Java Object, which will be managed by spring container through configuration. Service client will be able to consume SOAP service over HTTP/HTTPS. Booking information will be processed through this channel.

*Technology Benefit*:

* *Service client will be solely responsible for service consumption*
* *Service client can be configured through scheduler*
* *Separate component ensure responsibility separation*
* *Application maintenance and enhancement will be easier*

**Queue Client**

* Queue Client will be developed as Java Object. Queue clients will be used to listen to preconfigured configured queue. This will receive the data feed from external interfaces. After receiving the data feed. Data will be parsed, processed and uploaded into application database using Data Cleansing and Data Loading feature. flight information and changes will be received and processed through this channel.

*Technology Benefit*:

* *Queue client will be solely responsible for Message consumption through Queue*
* *Separate component ensure responsibility separation*
* *Application maintenance and enhancement will be easier*

**Application Security**

* We have assumed that this application will be integrated and launched from existing enterprise application. Hence existing enterprise Application authentication and authorization will be inherited into this application. Existing user roles and responsibility will be applicable for this application. Enterprise SSO policy will be implemented for this application also. Details of SSO integration will be done during application design phase.