Korean food restaurant

Software Architecture Document

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/10/2017 | 1.0 | Software Architecture Document generated using Rational SoDA template and Rational Rose model. | Thuy Duong  Minh Hien  Duc Huy |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms, and Abbreviations 3

1.4 References 3

1.5 Overview 3

2. Architectural Representation 3

3. Architectural Goals and Constraints 3

4. Use-Case View 3

4.1 Use-Case Realizations 3

5. Logical View 3

5.1 Overview 3

5.2 Architecturally Significant Design Packages 3

6. Process View 3

7. Deployment View 3

8. Implementation View 3

8.1 Overview 3

8.2 Layers 3

9. Data View (optional) 3

10. Size and Performance 3

11. Quality 3

Software Architecture Document

# Introduction

* This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of Korean food website. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the system.

## Purpose

* This document provides an architectural overview of Korean food website
* The primary purpose of this website is to manage restaurant, to access customers and to extend business
* This document is intended to capture and convey the significant architectural decisions which have been made in designing and building the website. It is a way by which developers and others involved in the project can better understand the problems to be solved and how it will be represented with this website. In order to depict the system as accurately as possible, the structure of this document is based on the “4+1” model view of architecture



## Scope

* This Software Architecture Document provides an architectural overview of Korean food website.

## Definitions, Acronyms, and Abbreviations

**RUP**: Rational Unified Process

**UML:** Unified Modeling Language

**SAD:** Software Architecture Document

## Overview

* This document consists of 7 sections, which are described below:
* \* Section 1 is simply an introduction to the software architecture of Korean food website
* \* Section 2 addresses the goals and constraints of the system’s architecture
* \* Section 3 describes the architectural representation of the system.
* \* Section 4 describes the five views in which the system documentation is divided by following the Rational Unified Process (RUP). A visual representation of these five views can be seen in the Software System Development Diagram.
* \* Section 5 of this document talks about other system considerations such as size and performance of the system.
* \* Section 6 describes some system quality issues.
* \* Section 7 is a bibliography of the references used to create this document.

# Architectural Representation

**Use Case view**

Audience: all the stakeholders of the system, including the end-users.

Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the software. Describes the actors and use cases for the software, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

Related Artifacts: Use-Case Model, Use-Case documents.

**Deployment view**

Audience: Deployment managers.

Area: Topology: describes the mapping of the software onto the hardware and shows the software's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

Related Artifacts: Deployment model.

**Logical view**

Audience: Designers.

Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.

Related Artifacts: Design model

**Process view**

Audience: Integrators.

Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.

Related Artifacts: (no specific artifact).

**Implementation view**

Audience: Programmers.

Area: Software components: describes the layers and subsystems of the application.

Related Artifacts: Implementation model, components

**Data view (optional)**

Audience: Data specialists, Database administrators

Area: Persistence: describes the architecturally significant persistent elements in the data model

Related Artifacts: Data model.

# Architectural Goals and Constraints

* This section describes the system requirements and objectives that have some significant impact on the architecture.
* Technical Platform
* Transaction
* .Net platform already has built in transaction capabilities, they will be used.
* Security
* The system must be secured, so that a customer can make online payments (Premium Membership)
* Basic security behaviors:
* Authentication: Login using at least a user name and a password
* Authorization: according to their profile, online user must be granted or not allowed to receive some specific services (Automatic match finding, Ride Suggestion, etc...)
* For internet access, the following requirements are mandatory
* Confidentiality: sensitive data must be encrypted if any (credit card payments).
* Safety: Credit card data must not be kept at a local database.
* Data integrity: Data sent across the network cannot be modified by a tier
* Auditing: Every sensitive action can be logged
* Non-repudiation: gives evidence a specific action occurred
* .NET security model will be reused

**Persistence**

Data persistence will be addressed using a relational database and .NET s Object Relational Mapping capability will be reused.

# Use-Case View

* This section provides a functional overview of the system by a use-case diagram.

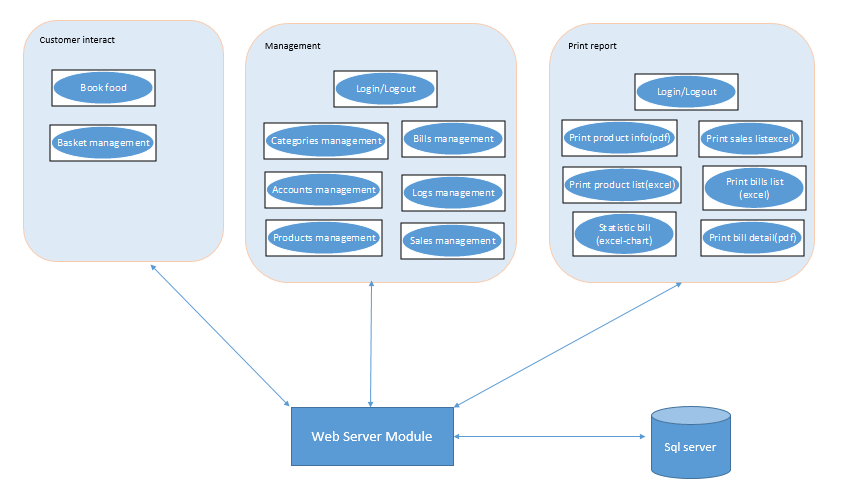
This is a list of use-cases that represent major functionality of the final system [SRS]:

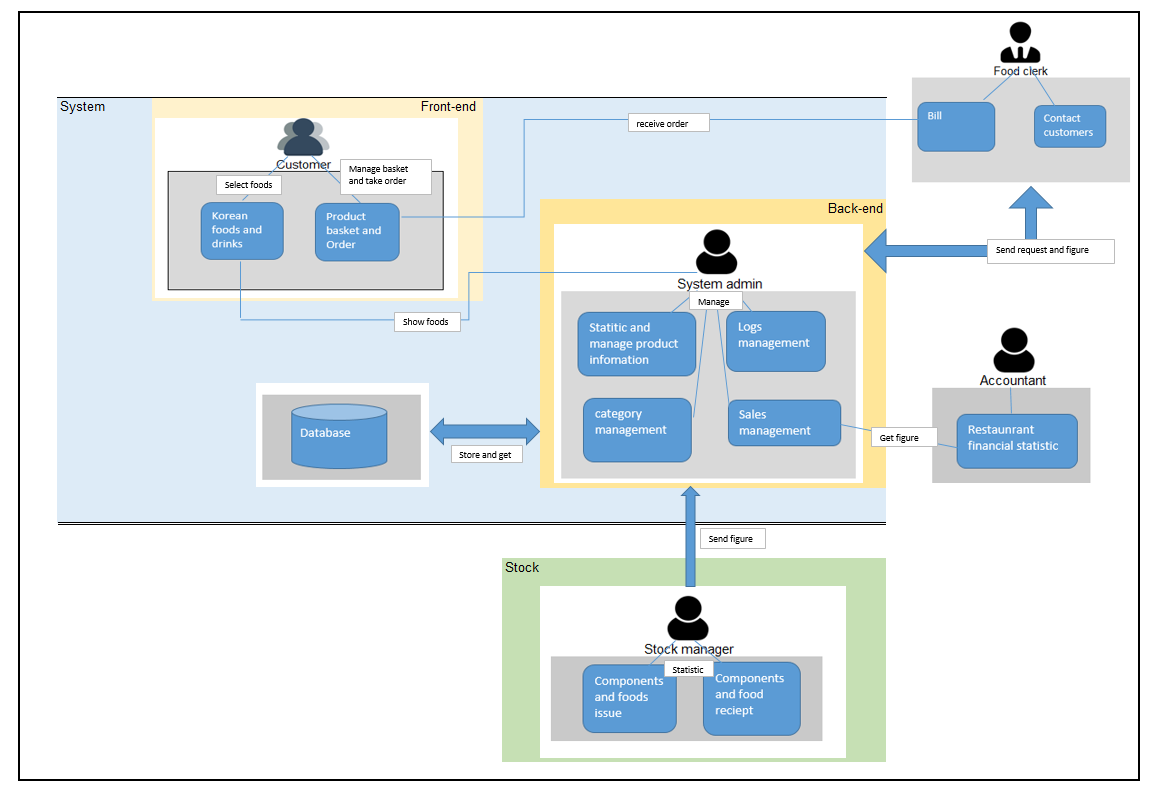
* Display foods
* Book foods
* Basket management
* Login, logout
* Account management
* Bill management
* Print bill
* Statistic bill
* Categories management
* Products management
* Print products list and detail
* Statistic products in stock
* Sales management
* Statistic sales
* Log management
* Print log



## Use-Case Realizations

Use case functionality diagram below describes how design elements provide the functionalities identified in the significant use-cases. Use cases are displayed as functionalities for the system. Functionality may enclose more than one use-case.





# Logical View

## Overview

DTCPII tool is divided into layers based on the N-tier architecture [KRU41].

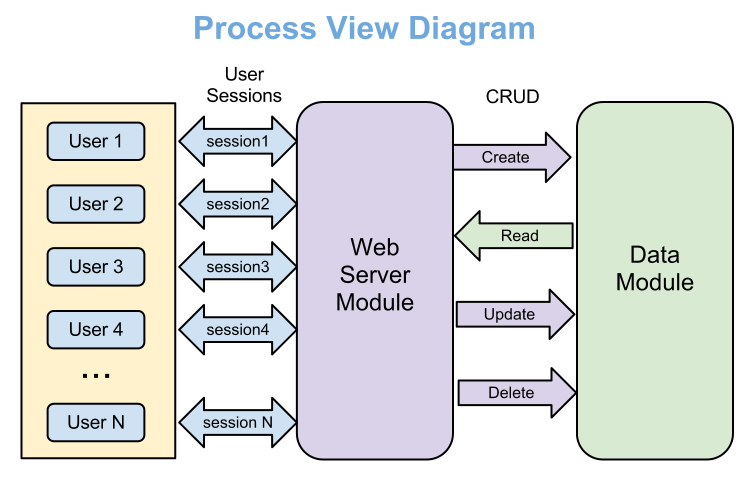
* 

The layering model of the DTCPII online application is based on a responsibility layering strategy that associates each layer with a particular responsibility.

This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.

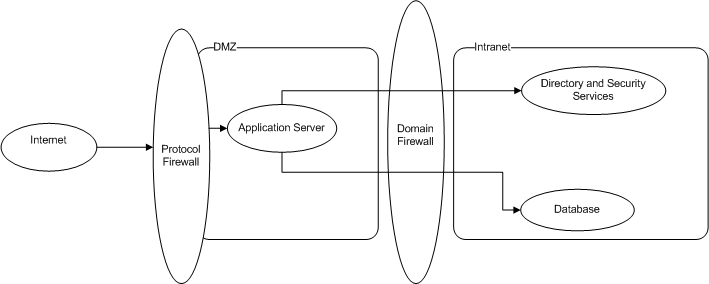
# Process View

Due to disconnected nature of HTTP request / response and ability of relational database management system (RDMS), DHCPII tool will handle multiple users simultaneously. Therefore, concurrency issues such as synchronous versus asynchronous mechanisms will be not considered in this document.

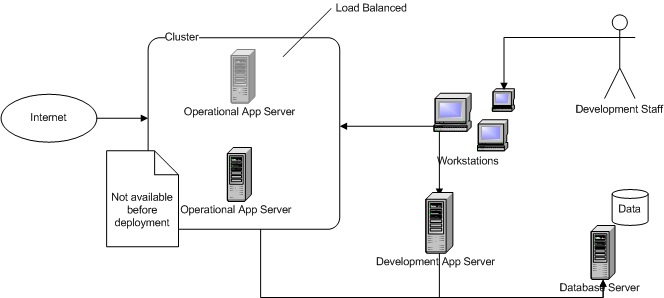
* 
* User – Creator, Reader or Administrator
* Session – HTTP session assigned by web server automatically
* CRUD – Create-Read-Update-Delete

# Deployment View

* Logical Structure



**Physical structure**

****

* Details
* Runtime Pattern is applied.
* Development is done on an extra application server
* Application is deployed into 2 application servers working in a cluster. (4 in the future)
* Application servers are IBM servers running IIS Application Server
* Database server is IBM server running MySQL server.
* All application servers and Database server have redundancy using mirroring RAID mode.
* Workstations run Linux and use Eclipse platform for development.

# Data View (optional)

The data view represents significant part of the DHCPII tool. Modularization (normalization) is selected as design approach of physical data model. Data consistency and quality are enforced through the series of Primary and Foreign Key constrains.

* Data access will be provided only through the user web interface, however Process Template tables (Templates, Components and Subcomponents) will be filled manually since we are not planning to create special front-end interface for that. Nevertheless, the Data View structure will allow easy maintainable because all process complexity is hidden in the template tables, therefore creating or modifying process template will require minimum efforts*.*



# Quality

Scalability

* Description: System’s reaction when user demands increase
* Solution: System is built to support 10000 members, further scalability is supported by new hardware and .NET

Reliability, Availability:

* Description: Transparent failover mechanism, mean-time- between-failure
* Solution: .NET application server supports load balancing through clusters, RAID mirroring is used diminish software crashes.

Portability:

* Description: Ability to be reused in another environment
* Solution: The system is fully .NET compliant and thus can be deployed onto any .NET application server

Security:

* Description: Authentication and authorization mechanisms
* Solution: .NET native security mechanisms will be reused