

BAKERY STORE

Report #2 – Architecture Design

|  |  |  |
| --- | --- | --- |
| **Bakery Store** | | |
| **Group member** | Phạm Thị Minh Phương | SE03773 |
| Cao Anh Tuấn | SE03737 |
| Hoàng Anh Phương | SE03715 |
| Ngô Trung Kiên | SE03736 |
| Nguyễn Xuân Hà | SE03484 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | BS | |

**Hà Nội, 09/2017**

Record of change

\*A - Added M - Modified D - Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\* M, D | Change Description | New Version |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

SIGNATURE PAGE

AUTHOR: Phạm Thị Minh Phương 15/09/2017

Project Manager

REVIEWERS: Cao Anh Tuấn 16/09/2017

Team member

APPROVAL: Nguyễn Văn Sang --/--/--17

Supervisor

TABLE OF CONTENTS

[1 Introduction](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc3871)

[1.1 Purpose](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc430)

[1.2 Scope](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc5436)

[1.3 Definitions, Acronyms and Abbreviations](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc24979)

[1.4 Overview](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc19836)

[2 CHOICE OF Architectural DESIGN](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc9511)

[2.1 MVC Model](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc24929)

[2.2 Node.js](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc25263)

[3 Architectural Goals and Constraints](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc32195)

[4 Use-Case View](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc19645)

[4.1 Administrator Group Function](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc18330)

[4.2 Staff Group Function](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc1622)

[4.3 Customer Group Function](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc18460)

[5 Logical View](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc9485)

[5.1 Overview](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc4828)

[5.2 Architecturally Significant Design Packages](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc14645)

[6 Process View](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc15820)

[7 Deployment View](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc13394)

[8 Implementation View](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc16970)

[8.1 Overview](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc2168)

[8.2 Layers](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc22433)

[9 Data View (optional)](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc28471)

[10 Size and Performance](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc6834)

[11 Quality](file:///C:\\Users\\PhamPhuong\\Desktop\\New%20folder%20(3)\\BS_Architecture-Design_EN-1.docx" \l "_Toc32061)

# Introduction

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

## Purpose

The scope of this document is to depict the architecture of the Bakery Store website created by BS capstone project team.

## Scope

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Definition** | **Note** |
| BS | Bakery Store |  |
| DB | Database |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |

## Definitions, Acronyms and Abbreviations

* BS\_ Software Requirements Specification\_v1.0\_EN.docx
* BS\_Data Design\_v1.0\_EN.docx
* Software Architecture Design Illuminated Book

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

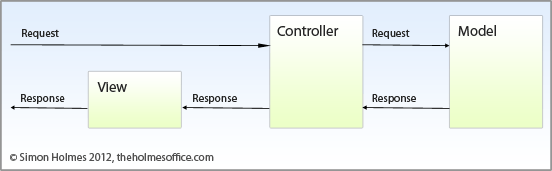
# CHOICE OF Architectural DESIGN

## MVC Model

### MVC Model Overview

A very popular approach for web sites & web apps – and the one we’ll be working with – is MVC, which stands for model, view, controller. This makes distinctions between the model (data), view (presentation) and controllers (logic).

The diagram below shows how a standard MVC request/response loop looks:



* **Controllers:** Controller is a level which acts like a brain of the entire MVC system.. A controller also acts as a link between a user and the system. It provides the user with input by providing appropriate views to present it appropriately on the screen. The controller understands user output, converts it into the appropriate messages and passesthe same to views.
* **Models:** This level is very important as it represents the data to the user. This level defines where the application’s data objects are stored. The model doesn’t know anything about views and controllers. So, whenever there are changes done in the model it will automatically notify observers that the changes are made. The model may be a single object or a structure of objects.
* **Views:** A view is a visual representation of the MVC model. This level creates an interface to show the actual output to the user. However, a view will not display anything itself. It is the controller or model that tells view what to display to the user. It also handles requests from the user and informs controller. A view is connected to its model and gets the data necessary for the presentation by asking certain questions. Sometimes, it also updates the model by sending appropriate messages. All these questions and messages aresent back to the model in such an easy terminology that it can easily understand the information sent by model or a controller.

### Advantage and disadvantage of MVC Model

* **Advantage:**
  + **Simultaneous development – Multiple developers can work simultaneously on the model, controller and views..**
  + **Modification does not affect the entire model:** Modification does not affect the entire model because model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture..
  + **MVC model returns the data without formatting:** MVC pattern returns data without applying any formatting so the same components can be used and called for use with any interface
  + **Ease of modification :** Because of the separation of responsibilities, future development or modification is easier.
* **Disadvantage:**
  + For small projects that apply MVC model caused cumbersome, time consuming in development process.
  + Time consuming to transits data between components.
  + The MVC model takes time to learn and apply properly.

### The reason for choosing MVC Model

* Better support for test-driven development (TDD)
* Helps to maintain, move, and organize applications more easily.
* Allow modification of code without affecting much of the other parts.

## Node.js

### Node.js Overview

Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/" \t "https://www.tutorialspoint.com/nodejs/_blank) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

### Advantages and disadvantages of Node.js

* Advantages:
* Node.js is an open-source framework under MIT license. (MIT license is a free software license originating at the Massachusetts Institute of Technology (MIT).)
* Uses JavaScript to build entire server side application.
* Lightweight framework that includes bare minimum modules. Other modules can be included as per the need of an application.
* Asynchronous by default. So it performs faster than other frameworks.
* Cross-platform framework that runs on Windows, MAC or Linux
* Disadvantages:
* It doesn’t support multi-threaded programming.
* It doesn’t support very high computational intensive tasks. When it executes long running task, it will queue all the incoming requests to wait for execution, since it follows JavaScript event loop which is single threaded.
* Node good for executing synchronous and CPU intensive tasks.

### The reasons for choosing Node.js

* **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping.Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.
* **License** − Node.js is released under the [MIT license](https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE" \t "https://www.tutorialspoint.com/nodejs/_blank)

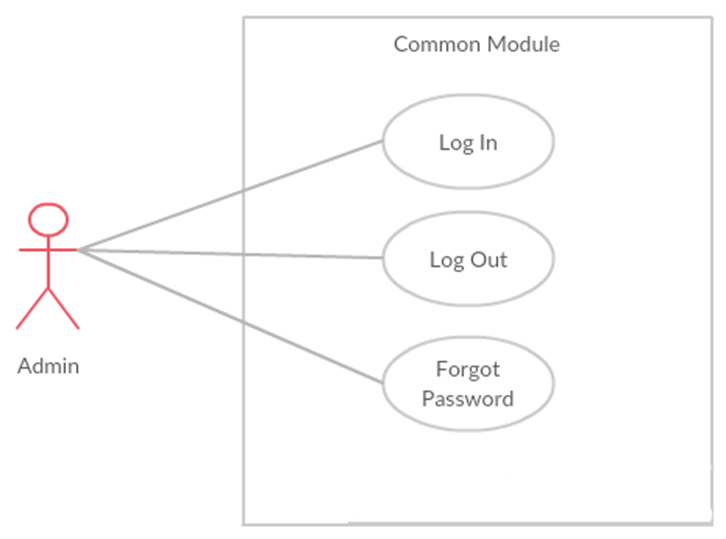
# Architectural Goals and Constraints

* **Availability:**
* The application must be available 95% of time. Users can access to it everywhere from there .Web browser with internet connection.
* **Maintainability:**
  + Coding standards and naming conventions:
    - Output of the project must include coding standards and naming conventions documentations. Implementation code must be easy to maintain.
    - All code must be clearly commented, including class, method documentations.
    - If some components are reused, the documentations of those components must also be included.
* Design:
  + - The design of the system must be loosely coupled that chances on some module will not affect others.
* Logging:
  + - All the errors should be logged, supporting for bug fixing and maintenance.
    - All strange or sensitive situations should also be logged.
* **Usability:**
  + Intuitiveness: all help/error messages are simple to understand; user can know exactly how to do each feature after one time using it.
* **Capacity and scalability:**
  + Throughput, storage and growth requirements.

# Use-Case View

## Administrator Group Function

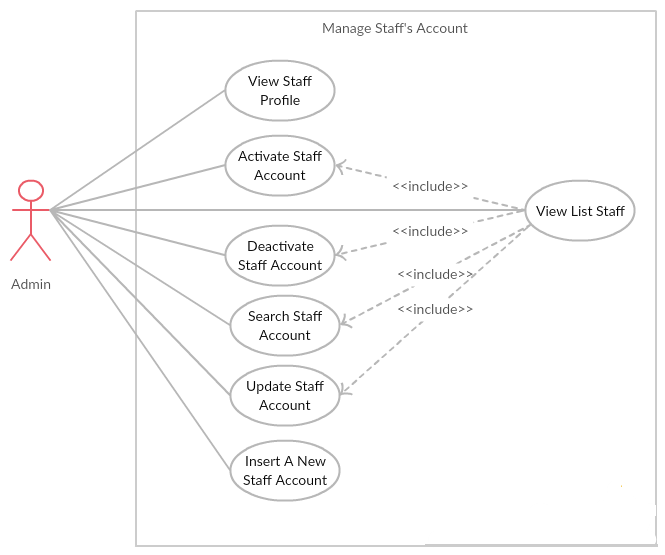
### Common Module



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Admin Login | Administrator | This function allows Admin login into website |
| 2 | Admin Logout | Administrator | This function allows Admin logout of BS website |
| 3 | Admin Forgot Password | Administrator | This function allows Admin reset password by email |

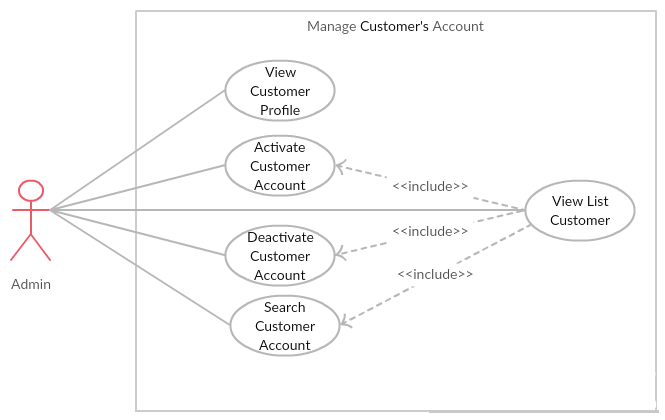
Table 4-1: Common Module US

### Manage Staff



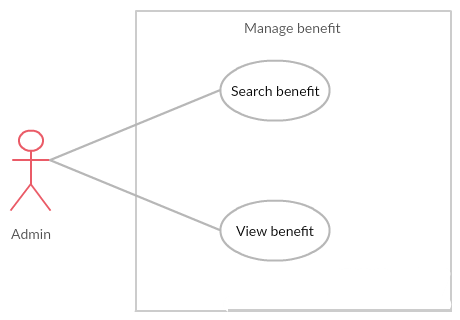
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | View list staff | Administrator | This function allows Administrator to view list staff of BS |
| 2 | View staff profile | Administrator | This function allows Administrator to view staff profile of BS |
| 3 | Activate staff account | Administrator | This function allows Administrator to Activate a staff’s account |
| 4 | Deactivate staff account | Administrator | This function allows Administrator to Deactivate a staff’s account |
| 5 | Search staff account | Administrator | This function allows Administrator to search a staff’s account |
| 6 | Update staff account | Administrator | This function allows Administrator to update a staff’s account |
| 7 | Insert a new staff account | Administrator | This function allows Administrator to insert a new staff account |

### Manage customer



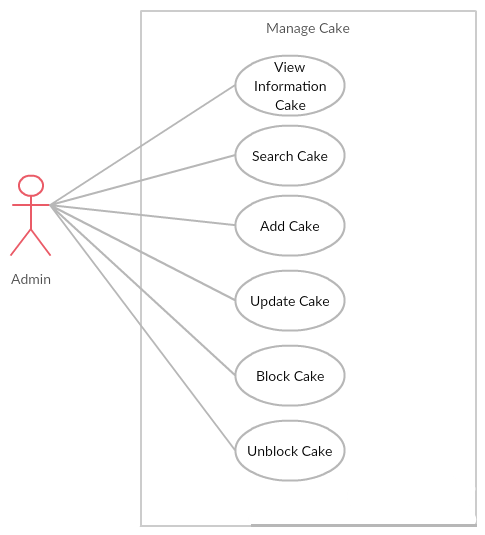
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | View list customer | Administrator | This function allows Administrator to view list customer of BS |
| 2 | Activate customer account | Administrator | This function allows Administrator to Activate a customer’s account |
| 3 | Deactivate customer account | Administrator | This function allows Administrator to Deactivate a customer’s account |
| 4 | View customer profile | Administrator | This function allows Administrator to view information of customer’s account |
| 5 | Search customer account | Administrator | This function allows Administrator to search a customer’s account |

### Manage benefit



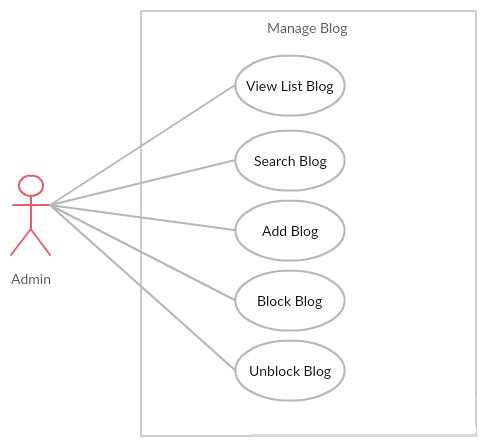
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Search benefit | Administrator | This function allows Administrator to search benefit by date of BS |
| 2 | View benefit | Administrator | This function allows Administrator to view detail benefit of BS |

### Manage cake



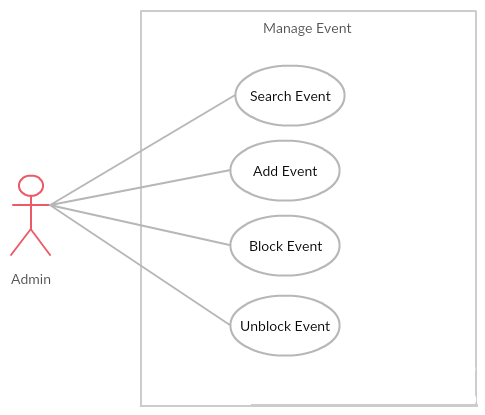
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Search cake | Administrator | This function allows Administrator to search cake of project |
| 2 | Add cake | Administrator | This function allows Administrator to add cake of project |
| 3 | Update cake | Administrator | This function allows Administrator to update cake of project |
| 4 | Block Cake | Administrator | This function allows Administrator to block cake will not sale of project |
| 5 | Unblock Cake | Administrator | This function allows Administrator to unblock cake will sale of project |
| 6 | View information cake | Administrator | This function allows Administrator to view detail of cake |

### Manage blog



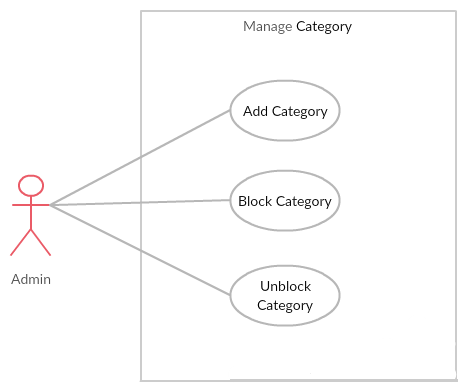
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Search blog | Administrator | This function allows Administrator to search blog of project |
| 2 | Add blog | Administrator | This function allows Administrator to add blog of project |
| 3 | Block blog | Administrator | This function allows Administrator to block blog of project |
| 4 | Unblock blog | Administrator | This function allows Administrator to Unblock blog of project |

### Manage event



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Search event | Administrator | This function allows Administrator to search event of project |
| 2 | Add event | Administrator | This function allows Administrator to add event of project |
| 3 | Block event | Administrator | This function allows Administrator to block event of project |
| 4 | Update event | Administrator | This function allows Administrator to update event of project |

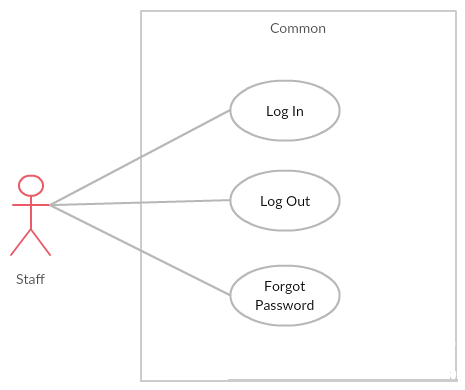
### Manage Category



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Add Category | Administrator | This function allows Administrator to add new category of project |
| 2 | Block Category | Administrator | This function allows Administrator to block category of project |
| 3 | Unblock Category | Administrator | This function allows Administrator to unblock category of project |

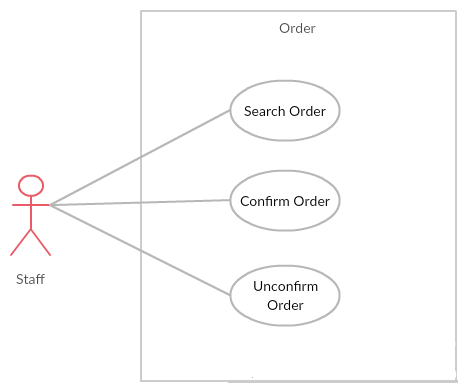
## Staff Group Function

### Common Module



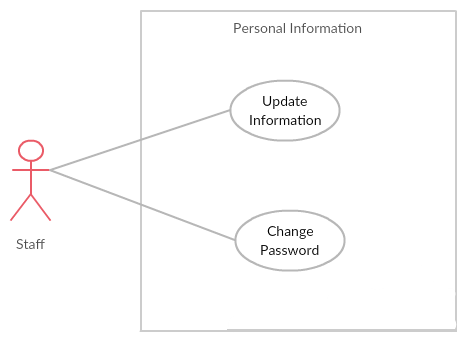
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Login | Staff | This function allows Staff login into website |
| 2 | Logout | Staff | This function allows Staff logout of BS website |
| 3 | Forgot password | Staff | When Staff forgot their password to login the system, this function will receiving their password to email |

### Manage order



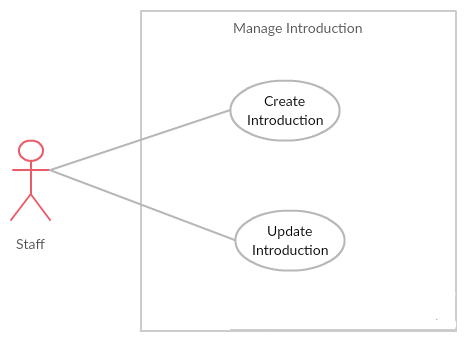
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Search order | Staff | This function allows Staff to search order of customer. |
| 2 | Confirm Order | Staff | This function allows Staff to confirm order transaction unsuccessful |
| 3 | Unconfirmed Order | Staff | This function allows Staff to unconfirmed order transaction successful |

### Personal Information



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Update Information | Staff | This function allows staff to edit profile |
| 2 | Change Password | Staff | This function allows staff to change password |

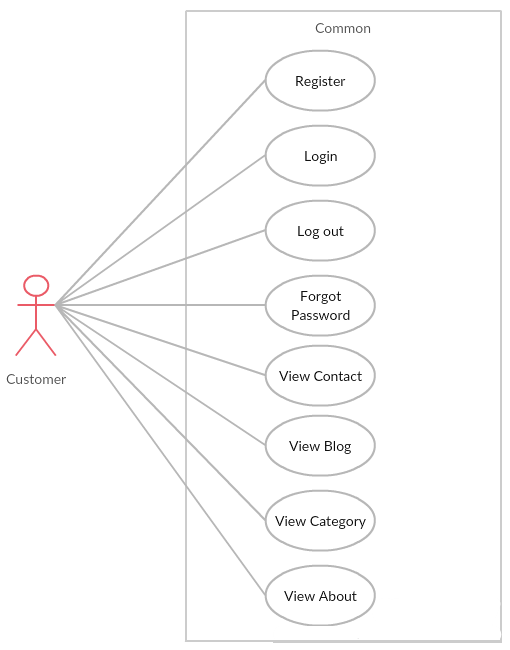
### 4.2.4Manage introduction



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Create introduction | Staff | This function allows Staff to create introduction of project |
| 2 | Update introduction | Staff | This function allows Staff to update introduction of project |

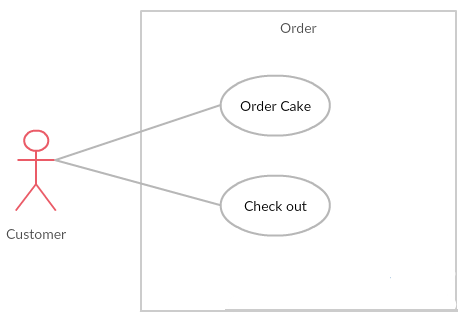
## Customer Group Function

### Common Module



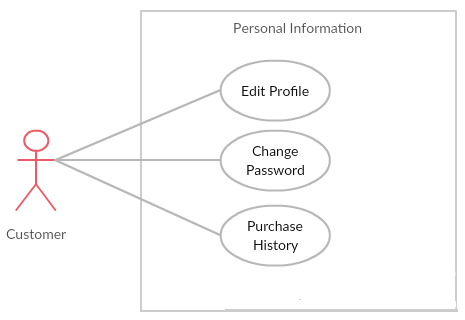
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Register | Customer | Create new account to use more function as a register user |
| 2 | Login | Customer | This function allows Customer login into website |
| 3 | Logout | Customer | This function allows Customer logout of BS website |
| 4 | Forgot password | Customer | When Customer forgot their password to login the system, this function will receiving their password to email |
| 7 | View blog | Customer | This function allows Customer View blog into website |
| 8 | View contact | Customer | This function allows Customer View contact into website |
| 9 | View category | Customer | This function allows Customer View category into website |
| 10 | View about | Customer | This function allows Customer View about into website |

### Order



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Order cake | Customer | This function allows Customer Order cake into website |
| 2 | Check out | Customer | This function allows Customer Check out |

### Personal Information



|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Use-case name** | **Actor** | **Description** |
| 1 | Edit Profile | Customer | This function allows customer to edit profile |
| 2 | Change Password | Customer | This function allows customer to change password |
| 3 | Purchase History | Customer | This function allows customer to purchase history |

# Logical View

## Overview

Logical View includes Package diagram and Class diagram. Package diagram describes the organization of packages and elements. Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system

* + Controller contain the interface between:
    - Associated models
    - Associated views
    - The input devices (e.g., keyboard, pointing device, time).
    - Send commands to the model to update the model's state.
  + Model is:
    - the domain-specific software simulation
    - Or implementation of the application's central structure.
  + View: deal with everything graphical:
    - Requests data from their model
    - Display the data.
  + Repository:
    - Create queries to DB.
    - Process data.
    - Return to controller.

## Architecturally Significant Design Packages

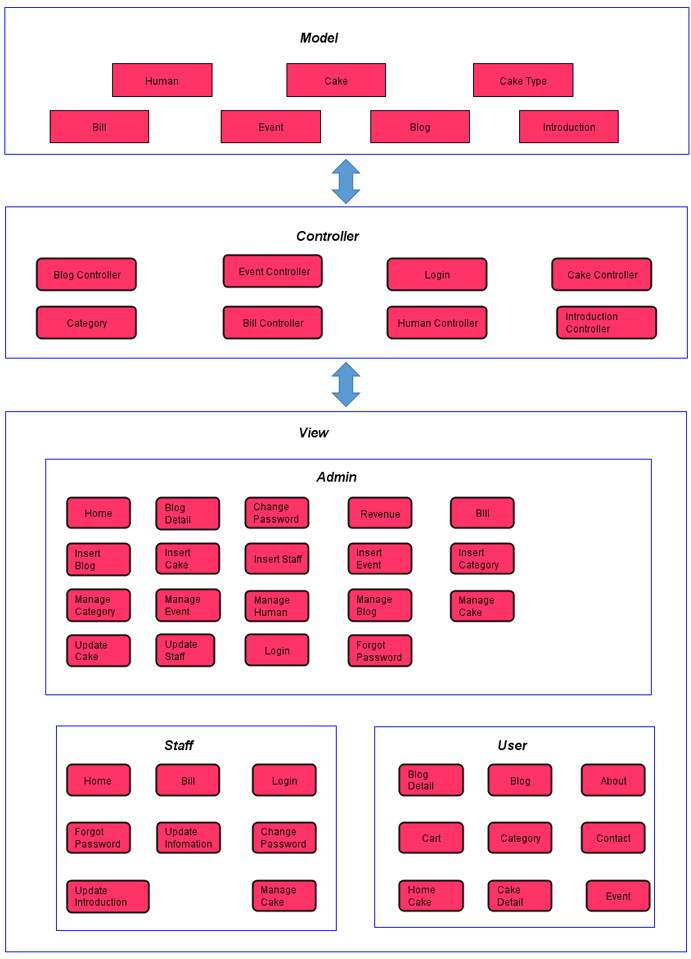


Figure 5‑1: Package Diagram

* **Model:**

|  |  |  |
| --- | --- | --- |
| **No** | **Model class** | **Role** |
|  | Human | Description entity of User in database |
|  | CakeType | Description entity of Category in database |
|  | Cake | Description entity of Cake in database |
|  | Event | Description entity of Event in database |
|  | Blog | Description entity of Blog in database |
|  | Bill | Description entity of Bill in database |
|  | Introduction | Description entity of Introduction in database |

Table 5‑1: Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| **No** | **Controller class** | **Role** |
|  | LoginController | * Receive request login, logout from client. * Call method login, logout. * Respond login view and login, logout status. |
|  | HomeController | * Receive request to home page from client. * Handle request from client and call methods to get all data of home page. * Respond data back to Home View. |
|  | AccountAdminController | * Receive requests (get, post, push, update) about Project from admin. * Respond status, message and data to client. |
|  | CakeAdminController | * Receive requests (get, post, push,update) about Project from admin. * Respond status, message and data to client. |
|  | BillController | * Receive request get admin partial views of from client. * Generate partial views and respond to client. |
|  | RevenueController | * Receive request get admin partial views of from client. * Generate partial views and respond to client. |
|  | ForgorPasswordController | * Receive requests from client and send email for client and respond message |
| **8** | OrderCakeController | * Receive requests from client and respond message |

* **Repository:**

|  |  |  |
| --- | --- | --- |
| **No** | **Repository class** | **Role** |
|  | BSDataContext | * Connect to DB, create queries to select, update, deletes data. |
|  | UserRepository | * Process user data and return to controller. |
|  | CakeRepository | * Process cake data and return to controller. |
|  | BlogResponsitory | * Process blog data and return to controller. |
|  | EventRepository | * Process event data and return to controller. |
|  | RevenueRepository | * Process revenue data and return to controller. |
|  | CategoryRepository | * Process category data and return to controller. |

* **DTOs:**

# Process View

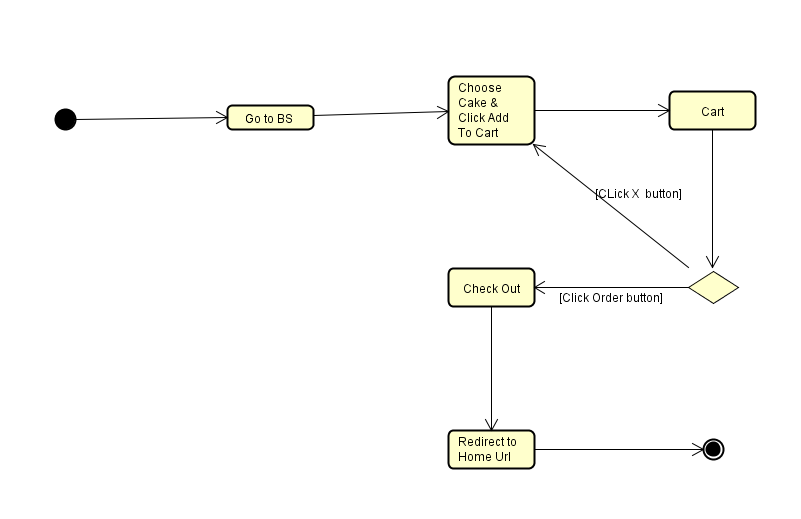


Figure 6-1: Check Out activity diagram

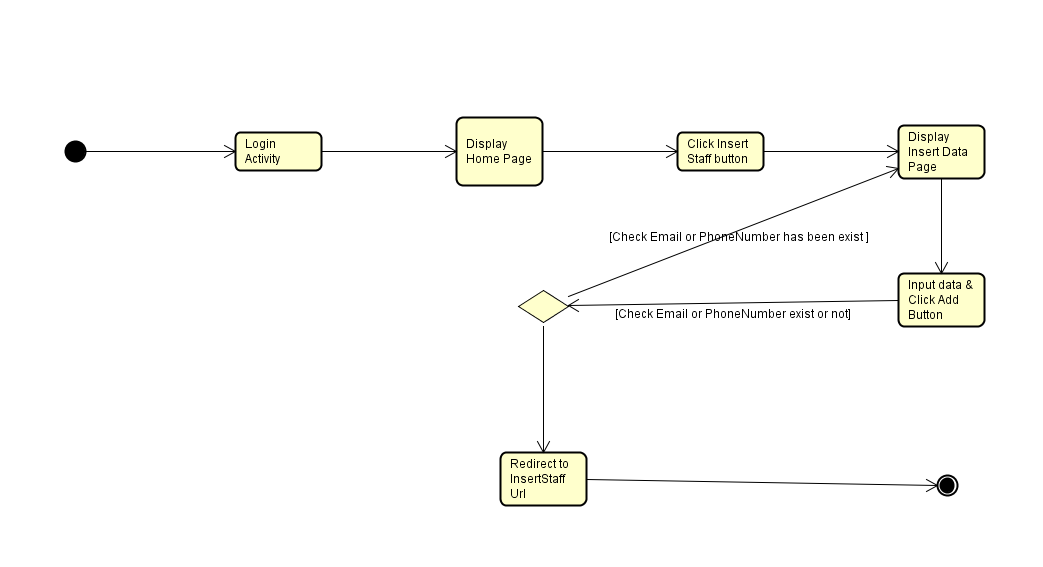


Figure 6-2: Insert staff activity diagram

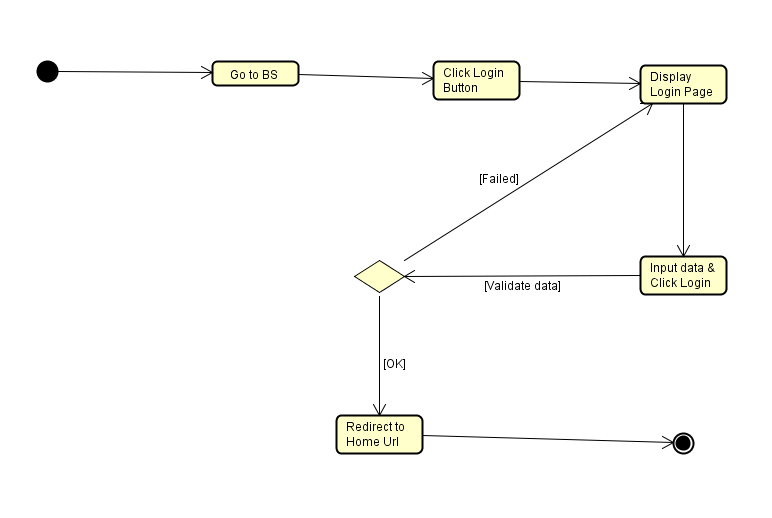


Figure 6-3: Log In activity diagram

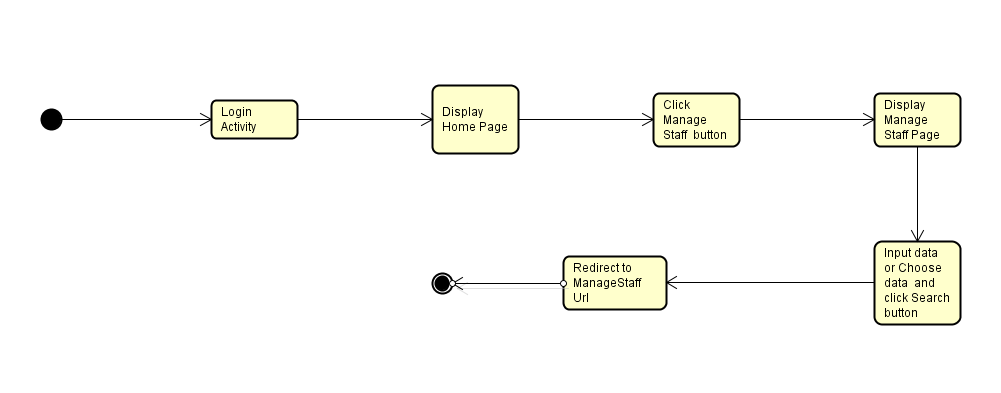


Figure 6-4: Search Staff activity diagram

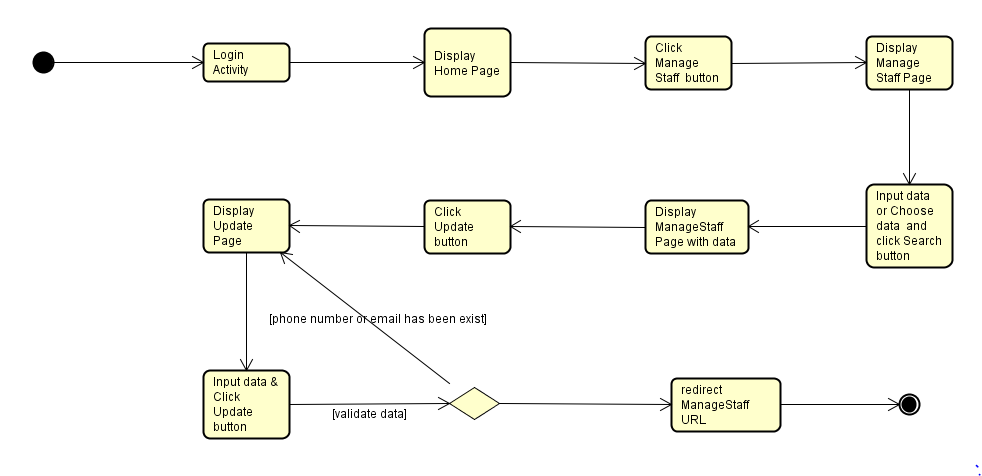


Figure 6-5: Update Staff activity diagram

# Deployment View

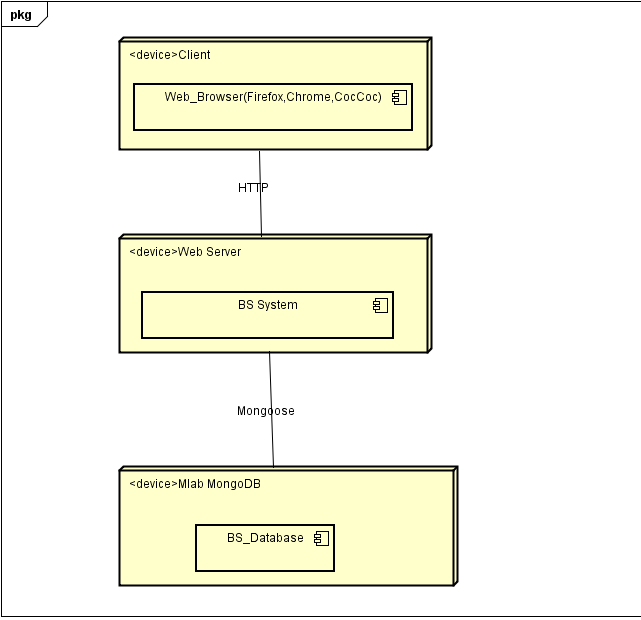


Figure 7-1: Deployment diagram

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | MongoDB | SQL Server use to store system’s data.  Using SQL Server 2010 or higher. |
| 2 | Client | Client is web browser to use system. Firefox 30, Chrome 40 or higher. |
| 3 | Web Server | Web server is hosted by… |

**Table 7‑1:** Deployment Diagram Description

# Quality

Reference to: BS\_Software Requirement Specification\_v1.0\_EN.docx