****

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**Car Rental Portal that manages the business of a bakery, only product for schools**

|  |  |
| --- | --- |
| **Group 1** | |
| **Group members** | Trần Hữu Đức – SE61448  Huỳnh Công Thành – SE61297  Lê Vũ Đăng Khoa – SE61238  Nguyễn Tường Tâm – SE61384 |
| **Supervisor** | Lâm Hữu Khánh Phương |
| **Ext. Supervisor** | N/A |
| **Capstone Project Code** | CRP |

- Ho Chi Minh City, 01 September 2016 -

This page is intentionally left blank

Table of Contents

[Table of Contents 3](#_Toc467165337)

[List of Tables 7](#_Toc467165338)

[List of Figures 8](#_Toc467165339)

[Definitions, Acronyms, and Abbreviations 9](#_Toc467165340)

[A. Introduction 10](#_Toc467165341)

[1. Project Information 10](#_Toc467165342)

[2. Introduction 10](#_Toc467165343)

[3. Current Situation 10](#_Toc467165344)

[4. Problem Definition 10](#_Toc467165345)

[5. Proposed Solution 11](#_Toc467165346)

[5.1. Featured functions 11](#_Toc467165347)

[5.2. Advantages 11](#_Toc467165348)

[5.3. Disadvantages: 11](#_Toc467165349)

[6. Functional Requirements 11](#_Toc467165350)

[7. Roles and Responsibility 12](#_Toc467165351)

[B. Software Project Management Plan 13](#_Toc467165352)

[1. Problem Definition 13](#_Toc467165353)

[1.1. Name of this Capstone Project 13](#_Toc467165354)

[1.2. Problem Abstract 13](#_Toc467165355)

[1.3. Project Overview 13](#_Toc467165356)

[2. Project organization 15](#_Toc467165357)

[2.1. Software Process Model 15](#_Toc467165358)

[2.2. Roles and responsibilities 16](#_Toc467165359)

[2.3. Tools and Techniques 17](#_Toc467165360)

[3. Project Management Plan 17](#_Toc467165361)

[3.1. Tasks 17](#_Toc467165362)

[3.2. Task sheet 20](#_Toc467165363)

[3.3. All Meeting Minutes 20](#_Toc467165364)

[4. Coding Convention 20](#_Toc467165365)

[C. Software Requirement Specification 21](#_Toc467165366)

[1. User Requirement Specification 21](#_Toc467165367)

[1.1. Guest requirements: 21](#_Toc467165368)

[1.2. Customer requirements: 21](#_Toc467165369)

[1.3. Provider requirements: 21](#_Toc467165370)

[1.4. Administrator requirements: 22](#_Toc467165371)

[2. System Requirement Specification 23](#_Toc467165372)

[2.1. External Interface Requirement 23](#_Toc467165373)

[2.2. System Overview Use Case 23](#_Toc467165374)

[2.3. Use Cases 25](#_Toc467165375)

[3. Software System Attribute 26](#_Toc467165376)

[3.1. Usability 26](#_Toc467165377)

[3.2. Reliability 26](#_Toc467165378)

[3.3. Availability 26](#_Toc467165379)

[3.4. Security 26](#_Toc467165380)

[3.5. Maintainability 27](#_Toc467165381)

[3.6. Portability 27](#_Toc467165382)

[3.7. Performance 27](#_Toc467165383)

[4. Conceptual Diagram 28](#_Toc467165384)

[D. Software Design Description 29](#_Toc467165385)

[1. Design Overview 29](#_Toc467165386)

[2. System Architecture Design 30](#_Toc467165387)

[2.1. Web Application architecture description 30](#_Toc467165388)

[3. Component Diagram 31](#_Toc467165389)

[4. Detailed Description 32](#_Toc467165390)

[4.1. Class Diagram 32](#_Toc467165391)

[4.2. Class Diagram Explanation 34](#_Toc467165392)

[4.3. Interactive Diagram 42](#_Toc467165393)

[4. User Interface Design 44](#_Toc467165394)

[5. Database Design 45](#_Toc467165395)

[6.1 Entity relationship diagram 45](#_Toc467165396)

[6.2 Entity Dictionary 47](#_Toc467165397)

[6. Algorithms 48](#_Toc467165398)

[E. システムの展開とテスト 49](#_Toc467165399)

[1. はじめに 49](#_Toc467165400)

[1.1 概要 49](#_Toc467165401)

[1.2 テストアプローチ 49](#_Toc467165402)

[2. データベース関係ダイアグラム 49](#_Toc467165403)

[2.1 ウェブアプリケーション物理的なダイアグラム 49](#_Toc467165404)

[2.2 ウェブアプリケーションデータ辞書 51](#_Toc467165405)

[3. パフォーマンス対策 51](#_Toc467165406)

[3.1 製品検索、メンバーのパフォーマンス 51](#_Toc467165407)

[3.2 送り状リストが展示されるパフォーマンス 51](#_Toc467165408)

[3.3 収入統計など展示されるパフォーマンス 51](#_Toc467165409)

[4. テスト計画 51](#_Toc467165410)

[4.1 テストする機能 51](#_Toc467165411)

[4.2 テストされない機能 52](#_Toc467165412)

[5. システムテストのテストケース 52](#_Toc467165413)

[F. ソフトウェアユーザーズマニュアル 53](#_Toc467165414)

[1. インストレーションガイド 53](#_Toc467165415)

[1.1. サーパ側の環境を設定 53](#_Toc467165416)

[1.2. サーパ側で展開 53](#_Toc467165417)

[1.3. クライアント側の環境の設定 57](#_Toc467165418)

[2. ユーザーガイド 58](#_Toc467165419)

[2.1. ログイン 58](#_Toc467165420)

[2.2. 製品を注文すること 59](#_Toc467165421)

[2.3. カートに製品を追加すること 59](#_Toc467165422)

[2.4. かーとをチェックすること 60](#_Toc467165423)

[2.5. カーとの精細 61](#_Toc467165424)

[2.6. 送り状を確認すること 62](#_Toc467165425)

[2.7. 送り状に顧客の情報を充実すること 62](#_Toc467165426)

[2.8. 送り状のリストを見ること 63](#_Toc467165427)

[2.9. 「人員」送り状を造ること 64](#_Toc467165428)

[2.10. 送り状に顧客を追加すること 65](#_Toc467165429)

[2.11. 新い顧客の口座で送り状をチェックアウト 66](#_Toc467165430)

[2.12. 入力法案 リスク 67](#_Toc467165431)

[2.13. 新入力法案を造ること 68](#_Toc467165432)

[2.14. 入力原料 リスクを見る 69](#_Toc467165433)

[2.15. 入力原料 の精細を見ること 70](#_Toc467165434)

[2.16. 新入力原料を造ること 71](#_Toc467165435)

[G. Appendix 73](#_Toc467165436)

List of Tables

List of Figures

Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Name | Definition |
| CRP | Car Rental Portal |

Table 1: Definitions, Acronyms, and Abbreviations

# Introduction

## Project Information

* Project name: **Car Rental Portal**
* Project Code: **CRP**
* Product Type: **Website Application**
* Start Date: **September 1st, 2016**
* End Date: **December 1st, 2016**

## Introduction

In this document, we introduce a car rental portal solution – a website application to help car rental service provider to bring their business online. The solution allows providers to define their own rental policy and price. It also simplifies the providers’ rental management process. Our solution also allows customer to easily find and book the most suitable car for their trip.

## Current Situation

Nowadays, there are many car rental providers in Vietnam. However, only a small percent of them have utilized internet to promote their business. Part of them (Large rental companies, taxi companies) have their own website or mobile application, while some choose to become driver for online rental portal solution like Uber and Grab.

Many small and middle-sized providers cannot afford their own website or mobile application, and also find contract with Uber and Grab too rigid because they are not allowed to define their own rental price and policy. Uber’s and Grab’s solutions also do not apply well for long-period rental.

A new solution that has just appeared recently is Aleka, which they themselves claimed to be inspired by Uber and Airbnb. Their approach take a lot after Airbnb. They allows providers to post their vehicles on their website, and then helps customer to find and book the most suitable vehicle in their system. Their solution however does not offer any way for customer to rate and comment about the service of provider, as well as to read other customers’ ratings and reviews before deciding to book the vehicle.

## Problem Definition

The current situation has several problems:

* **Cost barrier:** The cost to develop an online rental application is too high for small rental providers.
* **Ill-suited rental portal solutions:** There are great rental portal services like Uber, Grab, or Aleka on the market. However, their solutions have yet to completely satisfy the user. Their shortcomings include limiting provider’s action, cannot handle long-period rental very well, or lack of a mean for rating and reviewing provider’s service.

## Proposed Solution

Our proposed solution, named Car Rental Portal, is a car rental portal website application that takes after Airbnb. We provide a flexible mean for provider to promote their rental service online while still allows them to define their own rental policy. Our solution also allows customer to rate and comment about their past renting, as well as read others’ review on vehicles. Our solution also offers a searcher integrated with recommender engine that can help customer to find the best matched vehicle quickly and easily.

### Featured functions

* **Manage online rental service:** Provider can manage their vehicles’ and bookings’ information, as well as declare their own rental policies, including price contrainst, time constraint, travel distance constraint…
* **Advanced vehicle searcher with recommender engine:** System can help customers to find the most suitable vehicle fast and easily based on their booking history.
* **Review vehicle and read vehicle’s reviews:** Customer can comment and rate rental service after renting vehicle, as well as read reviews from other customer for a vehicle.
* **Book vehicle and cancel booking:** Customer can book and pay booking fee through NganLuong payment portal, as well as cancel their booking easily.

### Advantages

* Offer a cheap and easy online rental management solution.
* Allow provider to define their own rental policies.
* Support both hourly and daily rental.
* Offer vehicle searching with recommender engine.
* Offer rental service rating/reviewing function.
* Allow paying through NganLuong payment portal.

### Disadvantages:

* Since recommender works depending on customer’s booking history, customer must have booked at least once in the past to utilize it.
* Depends on NganLuong and has no alternative payment method in case NganLuong goes down.

## Functional Requirements

* Manage vehicles:
  + Provider can create, update, and delete vehicle.
* Manage vehicles’ bookings:
  + Provider can keep track of their vehicle’s booking schedule.
  + Provider can create self-booking to reserve their vehicle.
  + Provider can cancel self-booking.
* Manage garage:
  + Provider can create, update, and delete garage.
  + Provider can specify their garage’s rental policy.
  + Provider can group vehicle into garage.
  + Provider can close their garage temporary.
* Manage price group:
  + Provider can create, update, and delete price group.
  + Provider can group vehicle that he want to apply specific price options.
  + Provider can disable the group and every vehicle inside the group.
* Manage booking:
  + Customer can book vehicle.
  + Customer can cancel their booking.
  + Customer can keep track of their booking history.
  + Customer can rate and comment about their booking.
* Manage user account:
  + Administrator can lock and unlock user account.
* Search and view vehicle’s information.
* View report:
  + Administrator can view system’s business report in dashboard.
  + Provider can view personal business report in dashboard.
* Receive notification email:
  + Customer will receive notification email when he has booked a vehicle.
  + Provider will receive notification email when their vehicle has been booked.
  + Customer will receive notification email when he has canceled a booking.
  + Provider will receive notification email when their vehicle’s booking has been canceled.

## Roles and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Lâm Hữu Khánh Phương | Project Manager | Supervisor | phuonglhk@fpt.edu.vn |
| 2 | Trần Hữu Đức | Developer | Leader | ducthse61448@fpt.edu.vn |
| 3 | Huỳnh Công Thành | Developer | Member | thanhchse61297@fpt.edu.vn |
| 4 | Lê Vũ Đăng Khoa | Developer | Member | khoalvdse61238@fpt.edu.vn |
| 5 | Nguyễn Tường Tâm | Developer | Member | tamntse61384@fpt.edu.vn |

Table 2: Roles and Responsibility

# Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

* **Official name**: Car Rental Portal
* **Vietnamese name**: Cổng thông tin cho thuê xe
* **Abbreviation**: CRP

### Problem Abstract

Nowadays, there are many car rental providers in Vietnam. However, only a small percent of them have utilized internet to bring their business online. Aside from creating their own online rental applications, providers can also utilize rental solution services like Grab, Uber, or Aleka. These new services however still contain many flaws that keep a large number of providers away.

### Project Overview

#### Current Situation

Our project has the following problems:

* **Recommender engine is dependent on customer’s booking history:** The recommender engine cannot work on customer that has not placed any booking in the past.
* **Has no alternative payment method:** Our system depends completely on NganLuong for payment processing. There is currently no alternative payment method.

#### The Proposed System

The proposed system is a rental portal website application that can help car rental service provider to bring and manage their rental service online and allow customer to find and book vehicle easily. The system utilize NganLuongApi to process payment and has a recommender engine to improve vehicle searching.

Separated based on user’s role, the system offers the following functions:

* For guests:
  + Search vehicle.
  + View vehicle’s rental information.
* For customers:
  + Search vehicle.
  + View vehicle’s rental information.
  + Book vehicle and cancel booking.
  + Track booking history.
  + Register providership.
  + Receive notification email upon booking or canceling booking.
* For providers:
  + Manage vehicles.
  + Manage vehicles’ bookings.
  + Manage garages.
  + Manage price groups.
  + View personal business report.
  + Extend providership period.
  + Receive notification email upon receiving booking or booking was canceled.
* Administrator:
  + Manage user accounts.
  + View system’s business report.

#### Boundaries of the System

* The system targets mainly small and mid-sized car rental service providers, but can also handle the needs of large providers with large number of vehicles and multiple garages.
* The system supports both daily and hourly booking.
* The language of the system is Vietnamese.
* The completed product includes:
  + Booking system for customer.
  + Rental service managing system for provider.

#### Future Plans

* Offer pricing scheduler so that price group can change price during special events.
* Support self-driving rental.
* Support motorbike rental.
* Offer driver management functions.
* Support internationalization.
* Develop mobile version.

#### Development Environment

##### Hardware requirement

For web server:

| Hardware | Minimum Requirements | Recommended |
| --- | --- | --- |
| Network Bandwidth | 4 Mbps | 8 Mbps |
| Operating System | Window Server 2008 | Window Server 2008 |
| Computer Processor | Intel® Xeon Dual Core® 1.4GHz (4MB Cache) | Intel® Xeon® Quad Core (8M Cache, 2.40 GHz) |
| Computer Memory | 2GB RAM | 4GB or more |
| Hard disk | 40GB | 80GB |

Table 3: Hardware requirement for Web Server

For web development:

| Hardware | Minimum Requirements | Recommended |
| --- | --- | --- |
| Internet Connection | 512 Kbps | 8 Mbps |
| Operating System | Window Vista, 7, 8 | Window 7, 8 |
| Computer Processor | 1 GHz | Intel® Core™ i5 CPU 2.53 GHz |
| Computer Memory | 2GB RAM | 4GB or more |

Table 4: Hardware requirement for Web Server

##### Software requirement

| Software | Name / Version |
| --- | --- |
| Operating system | Windows 7 or above |
| Modeling tool | StarUML version 2.7.0 (UML2.0 standard [[2](#_Appendix)]) |
| IDE | Microsoft Visual Studio 2015 Community |
| DBMS | SQL Server 2014 |
| Source control | Git 2.9.0.windows.1 |
| Web browser | Chrome (v.42 and above), Firefox (v.38 and above), and any equivalent web browser |

Table 5 : Software requirement

## Project organization

### Software Process Model

Modified waterfall model (Sashimi [[1](#_Appendix)]) is applied in this project. This model was chosen because:

* The requirements of the project are well-defined and well-understood by all team members.
* The deliverables in each phase of the model can be corresponded with the reports that have to be submitted in each process of the capstone project.

Requirement

Design and Architecture

Development and Coding

QA and Testing

Maintenance

Figure 1: Modified Waterfall Development Model

### Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No | Full name | Role in Group | Responsibilities |
| 1 | Lâm Hữu Khánh Phương | Supervisor / Project Manager | - Clarify user requirement  - Give technical support and business analysis  - Track development process  - Review document and product |
| 2 | Trần Hữu Đức | Team leader, Business Analyst, Developer, Tester | - Manage development process  - Analyse requirement  - Design database  - Prepare documents  - Design GUI  - Code  - Test  - Deploy product |
| 3 | Huỳnh Công Thành | Team member, Business Analyst, Developer, Tester | - Analyse requirement  - Design database  - Prepare documents  - Design GUI  - Code  - Test |
| 4 | Lê Vũ Đăng Khoa | Team member, Business Analyst, Developer, Tester | - Analyse requirement  - Design database  - Prepare documents  - Design GUI  - Code  - Test |
| 5 | Nguyễn Tường Tâm | Team member, Business Analyst, Developer, Tester | - Analyse requirement  - Design database  - Prepare documents  - Design GUI  - Code  - Test |

Table 6: Roles and responsibilities

### Tools and Techniques

| Tool / Technique | Name / version |
| --- | --- |
| Frontend | HTML5, CSS3, JavaScript, jQuery, Bootstrap |
| Backend | ASP.NET MVC 5, Entity Framework, NganLuong, Cloudinary |
| Web server | IIS 8 |
| Development tool | Microsoft Visual Studio 2015 Community |
| DBMS | MSSQL Server 2014 |
| Source control | Git 2.9.0.windows.1 |
| Modeling tool | StarUML 2.7.0 |
| Document tool | Microsoft Word 2013 |

Table 7: Tools and Techniques

## Project Management Plan

### Tasks

#### Initiating

| Task name | Initiating |
| --- | --- |
| Description | - Clarify user requirements on the rental portal.  - Research business logic of car rental service.  - Research popular car rental solutions (Grab, Uber, Aleka) as well as other similar websites (Carmudi, Airbnb).  - Offer team’s solution, declare its pros and cons, introduce some core function of the new system. |
| Deliverables | Report No.1 – Introduction |
| Resource needed | 20 man-days |
| Dependencies and constraints | No |
| Risk | - Missing requirement  - Unclear project scope  - Lacking teamwork in new formed team |

#### Planning

| Task name | Planning |
| --- | --- |
| Description | - Review team’s solution  - Verify project’s scope  - Build project management plan |
| Deliverables | Report No.2 – Software Project Management Plan |
| Resource needed | 16 man-days |
| Dependencies and constraints | - Base on Report No.1 – Introduction.  - Planned project must be completed in the following 12 weeks |
| Risk | Impractical plan |

#### Specifying requirements

| Task name | Specifying requirements |
| --- | --- |
| Description | Identify and clarify software requirements. |
| Deliverables | Report No. 3 – Software Requirement Specification |
| Resource needed | 20 man-days |
| Dependencies and constraints | Base on Report No.2 – Software Project Management Plan. |
| Risk | Misunderstood or unclear system’s requirement |

#### Designing database

| Task name | Designing database |
| --- | --- |
| Description | - Create conceptual, logical and physical database designs  - Implement in SQL Server 2014 |
| Deliverables | Physical database and SQL script. |
| Resource needed | 12 man-days |
| Dependencies and constraints | Base on Report No. 3 Software Requirement Specification |
| Risk | Unreasonable database design |

#### Create Software Design Description

| Task name | Create Software Design Description |
| --- | --- |
| Description | Decide software architect and clarify software detail design. |
| Deliverables | Report No. 4 – Software Design Description |
| Resource needed | 40 man-days |
| Dependencies and constraints | Base on Report No. 3 Software Requirement Specification and designed database |
| Risk | Unreasonable software design |

#### Implementing

| Task name | Implementing |
| --- | --- |
| Description | Implements all functions of system. |
| Deliverables | Software package. |
| Resource needed | 80 man-days |
| Dependencies and constraints | - Base on Software Requirement Specification and Software Design Description.  - Follow coding convention. |
| Risk | Incorrect implementation |

#### Testing

| Task name | Testing |
| --- | --- |
| Description | - Create test plan  - Perform tests.  - Fix bugs |
| Deliverables | Report No. 5 – System Implementation & Test |
| Resource needed | 40 man-days |
| Dependencies and constraints | Implementation is finished |
| Risk | - Incompleted test plan  - Lacking time for iterative test and bug fixing |

#### Creating User’s Manual

| Task name | Creating User’s Manual |
| --- | --- |
| Description | Create user’s manuals |
| Deliverables | Report No.6 – Software User’s Manual |
| Resource needed | 12 man-days |
| Dependencies and constraints | Product has passed acceptance test. |
| Risk | Bad and unclear documenting |

### Task sheet

Refer to “Task sheet” folder in the attached CD.

### All Meeting Minutes

Refer to “Meeting minutes” folder in the attached CD.

## Coding Convention

This project follows “C# coding conventions”.

* Naming Convention :
  + - For variable’s name, use camel case (numOfVehicle, currentDate…)
    - For function name, class name, use pascal case (SearchVehicle, VehicleInfoViewModel…)
* Layout Convention :
  + - Use the default Code Editor Settings (smart indenting, four-character indents, tabs saved as spaces).
    - Write only one statement/declaration per line.
    - Add at least one blank line between method definitions and property definitions.
    - Use parentheses to make clauses in an expression apparent
* Commenting Convention :
  + - Place the comment on a separate line, not at the end of a line of code.
    - Begin comment text with an uppercase letter.
    - End comment text with a period.
    - Insert one space between the comment delimiter (//) and the comment text.
* Language Guidelines :

<https://msdn.microsoft.com/en-us/library/vstudio/ff926074.aspx>

# Software Requirement Specification

## User Requirement Specification

### Guest requirements:

A guest is a person who has not been authenticated by the system. A guest has access to a limited number of system’s functions, which are:

* Log in
* Sign up
* Recover account’s password
* Search vehicle
* View rental information of vehicle

### Customer requirements:

A customer is an authenticated user that can utilize the system to find and book vehicle. A customer is able to:

* Log out
* Change personal information
* Change account’s password
* Search vehicle
* View rental information of vehicle
* Book vehicle
* Cancel booking
* Track booking history
* Comment and rate the car provider’s service after:
  + The booking has been canceled
  + The rental period has started
* Receive notification email when:
  + A booking has been placed successfully using this account
  + A booking has been canceled using this account

### Provider requirements:

A provider is an authenticated user that can use the system to manage their car rental business. A provider can:

* Log out
* Change personal information
* Change account’s password
* Search vehicle
* View rental information of vehicle
* Manage vehicles:
  + Register vehicle
  + Deregister vehicle
  + Update vehicle’s information
  + Group vehicles into garage
  + Group vehicles into vehicle group for price management
* Manage vehicles’ bookings:
  + Review booking schedule
  + Create self-booking to reserve vehicle
  + Cancel self-booking
* Manage garages:
  + Register new garage
  + Deregister garage
  + Edit garages’ information
  + Close garage
  + Reopen garage
  + Add vehicle to garage
  + Move vehicle to another garage
* Manage vehicle grouping for easier price management:
  + Register new group
  + Deregister group
  + Change group’s information and pricing
  + Deactivate group
  + Reactivate group
  + Add vehicle to group
  + Remove vehicle from group
* Extend providership period
* Receive notification email when
  + A vehicle has been booked.
  + A customer has canceled a booking.
* View personal business report

### Administrator requirements:

An administrator is an authenticated user that is tasked with managing the system and its users. Administrator is able to:

* Log out
* Change personal information
* Change account’s password
* Search vehicle
* View rental information of vehicle
* Deactivate customer/provider account
* Reactivate customer/provider account
* View system’s business report

## System Requirement Specification

### External Interface Requirement

#### User interface

* Graphics User Interface (GUI) should be simple, clear, intuitive, and reminiscent.
* GUI uses Vietnamese language.
* GUI uses consistent palette of colors between the text and the background.
* GUI displays best on screen with 1366x768 or higher resolution.

#### Hardware Interface

* N/A

#### Software Interface

The website works well on Chrome (v.42 and above), Firefox (v.38 and above), and any equivalent web browser.

#### Communication Protocol

Web server supports HTTP protocol v.1.1.

### System Overview Use Case

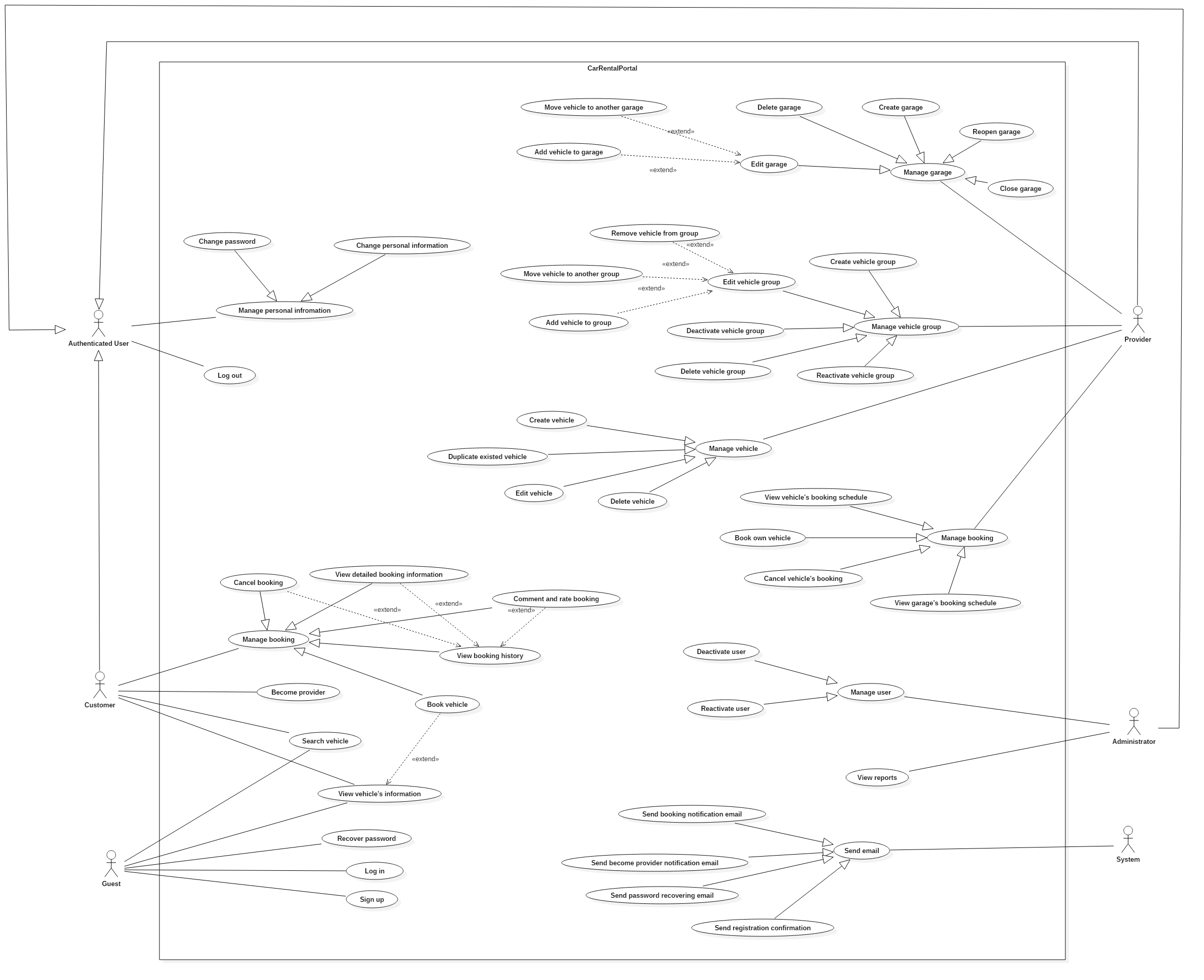


Figure 2: System Overview Use Case

### Use Cases

#### <Guest> Overview Use Case

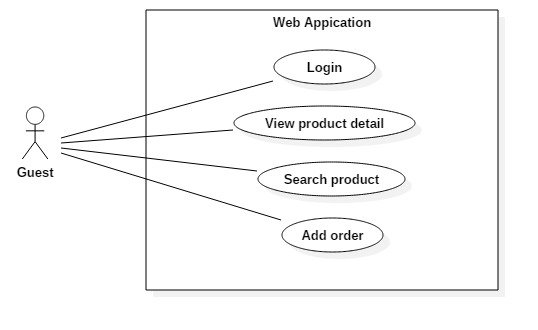


Figure 3: <Guest> Overview Use Case

#### <Authenticated user> Overview Use Case

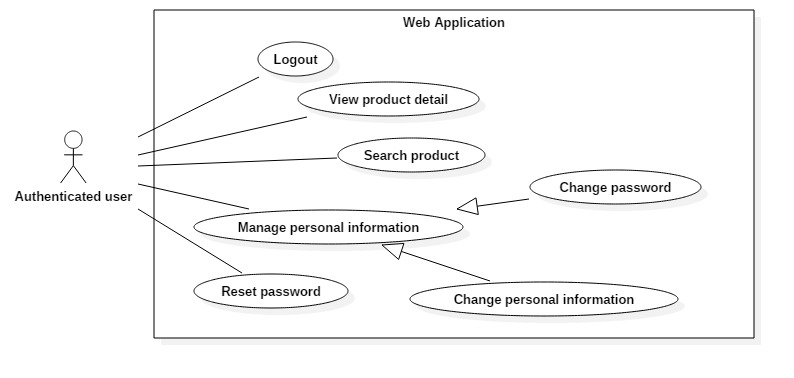


Figure 8: <Authenticated user> Overview Use Case

#### <Customer> Overview Use Case



Figure 17 <Customer> Overview Use Case

## Software System Attribute

### Usability

#### Graphic User Interface

GUI must use Vietnamese as main language and should be simple, clear, intuitive, and, reminiscent.

#### Usability

* Rental service provider should be able to operate the system properly in the matter of hours.
* The system must be simple and intuitive so that customer can start using without any taking any tutorial or training.

#### Installation

The system should be easy and fast to deploy following installation guide.

### Reliability

The system must not have any inconsistency in booking processing.

### Availability

The system should not have any downtime beside 6 hours of monthly maintenance.

### Security

* System must implement strict authorization and authentication.
* Passwords have to be encrypted before saving into database.

### Maintainability

* The system must be divided into separated, high-cohesive, and low-coupling modules.
* Code should be clear, DRY, and have helpful commends.

### Portability

* The web server can be deployed in any server machine that supports IIS8.
* Website should be able to run on Chrome (v.42 and above), Firefox (v.38 and above), and any equivalent web browser.

### Performance

Server must respond to user’s request from machine with minimum 4 Mbps connection and 1GHz CPU in less than 5 seconds.

## Conceptual Diagram

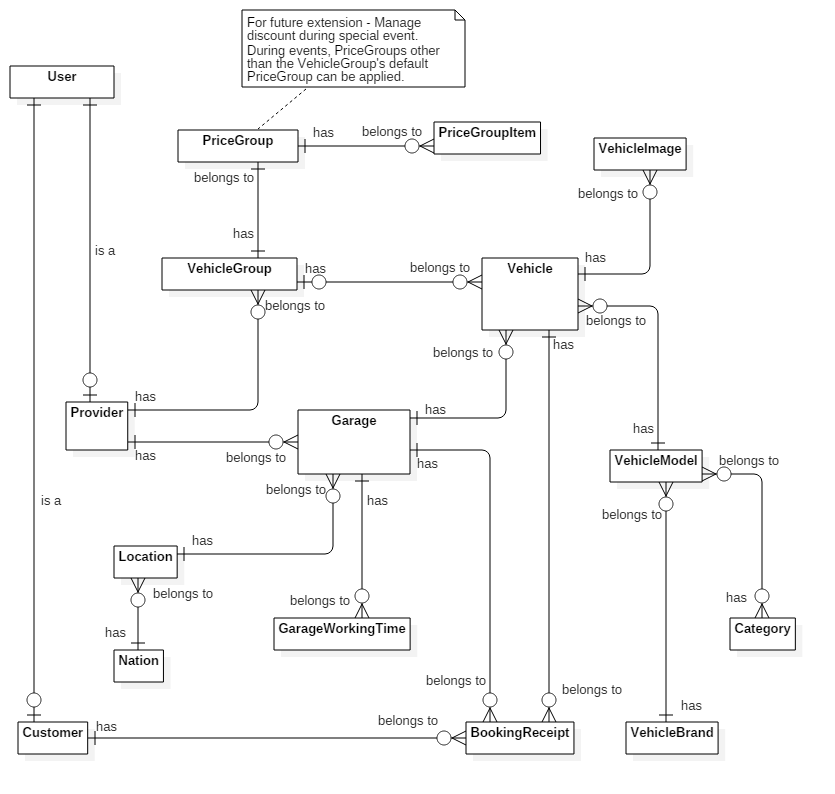


Figure 60: Conceptual Diagram

**Data Dictionary**

|  |  |
| --- | --- |
| Entity Data dictionary: describe all content of all entities | |
| Entity Name | **Description** |
| User | Abstract entity, represent user in system |
| Provider | User with provider role |
| Customer | User with customer role |
| Vehicle | Represent vehicle in system |
| VehicleImage | Represent image of vehicle |
| VehicleModel | Represent vehicle model |
| VehicleBrand | Represent vehicle brand |
| Category | Represent vehicle category |
| Garage | Represent garage |
| GarageWorkingTime | Describe working time constraints of garage each day of week |
| Location | Represent location inside a country |
| Nation | Represent nation |
| VehicleGroup | Represent group of vehicles with the same rental constraints |
| PriceGroup | Describe rental constraints of a vehicle group |
| PriceGroupItem | Describe rental constraints of each daily rental option of price group |
| BookingReceipt | Represent booking receipt in system |

# Software Design Description

## Design Overview

This document describes the technical and user interface design of CRP System. The document includes architectural design, detailed design of core functions and business functions, and database design.

The architectural design describes the overall architecture of the system and relations between components and sub-systems inside.

The detailed design describes static and dynamic structure of each component and function. The design includes class diagram, class explanations, and sequence diagrams of each of the important use case.

The database design describes the relationships between system entities and details about attributes of each entity.

**Document overview:**

* Section 2: System architectural design.
* Section 3: Component diagrams describes the connections and integration of components inside the system.
* Section 4: Detailed design description including class diagram, class explanations, and sequence diagrams.
* Section 5: User interface design.
* Section 6: Database design.
* Section 7: Algorithms.

## System Architecture Design

|  |
| --- |
| HTTP Request (1)    (5)  Object Relational Mapping (ORM)  Return data (7)  Query (6)  HTTP Response (11)  (10)  (3)  (8)  View Model  (9)  Model  (4)  Entity Framework 6.0  ADO.NET Data Provider  Database  Web browser  **Annotation**  Call / Send  Dependency  Select (2)  URL Routing  View  Business  Controller |

Figure 61: System architecture design

### Web Application architecture description

The system is developed using ASP.NET MVC5 and Entity Framework 6.0 (EF6). The system has 5 important parts:

* **Model** is the part that acts as a data transfer object between the system and database. Models are generated using Object relational mapping (ORM) offered by EF6.
* **View** is the part that handles the display of data. A view renders the display using data from the model or view model that is passed down by controller.
* **Controller** is the part that acts as event handler to handle the user interaction. Typically, a controller reads data from the request, calls the appropriate Business’s method, then selects view to render display and then send response back to user.
* **Business** is the part that contains all business handling methods.
* **View Model** is the part that acts like value object. Typically controller passes view model to view and view uses view model as data source for display rendering.

## Component Diagram

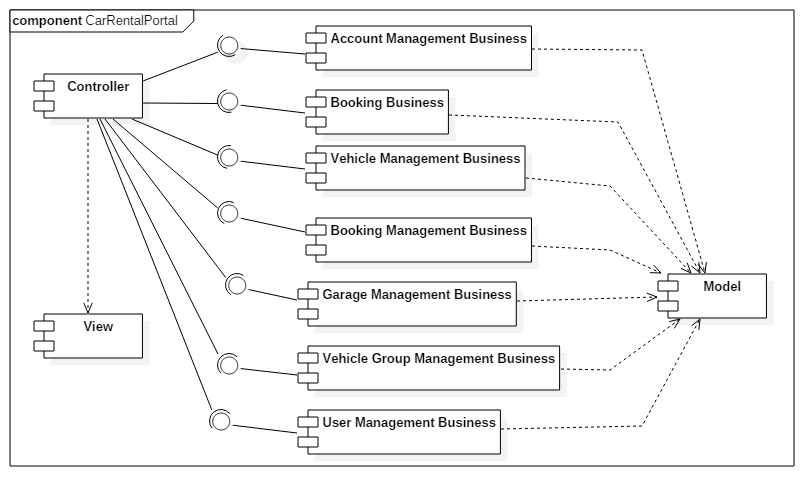


Figure 8 : Component diagram

|  |  |
| --- | --- |
| Component | Description |
| Controller | Represent all controllers in the system. |
| View | Represent all views in the system. |
| Model | Represent all models mapped by Entity Framework. |
| Account Management Business | Contain functions for handling account management (Register, change account information, change password …). |
| Booking Business | Contain functions for handling vehicle booking actions from customer. |
| Vehicle Management Business | Contain methods for handling vehicle management actions from provider. |
| Booking Management Business | Contain functions for handling booking management actions from provider. |
| Garage Management Business | Contain functions for handling garage management actions from provider. |
| Vehicle Group Management Business | Contain functions for handling vehicle group management actions from provider. |
| User Management Business | Contain functions for handling user management actions from admin. |

Table 8: Component dictionary

## Detailed Description

### Class Diagram

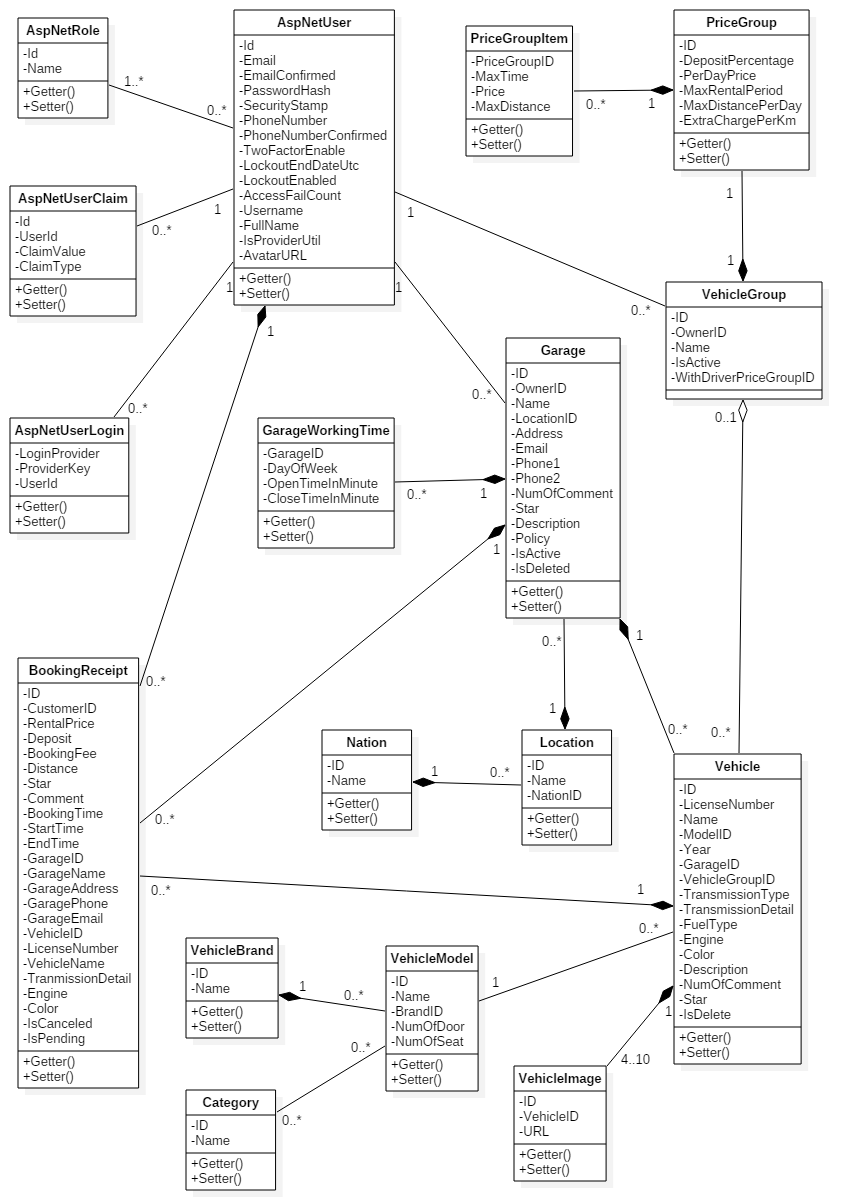


Figure 1: Class Diagram

|  |  |  |
| --- | --- | --- |
| Class dictionary | | |
| Class Name | **Mapping column with Conceptual diagram** | **Description** |
| AspNetUser | User | Generated by ASP.NET MVC5. Describe all user in system |
| AspNetRole | N/A | Generated by ASP.NET MVC5. Describe existing user’s roles in system. |
| AspNetUserClaim | N/A | Generated by ASP.NET MVC5. Describe all stored claims (Special attributes) of AspNetUsers. |
| AspNetUserLogin | N/A | Generated by ASP.NET MVC5. Describe information about 3rd party/external logins allowed for each AspNetUsers. |
| Vehicle | Vehicle | Describe all vehicles in system. |
| VehicleImage | VehicleImage | Describe all images of each vehicle. |
| VehicleModel | VehicleModel | Describe all vehicle models in system. |
| VehicleBrand | VehicleBrand | Describe all vehicle brands in system. |
| Category | Category | Describe all vehicle categories in system. |
| VehicleGroup | VehicleGroup | Describe all vehicle groups in system |
| PriceGroup | PriceGroup | Describe rental constraints (Pricing, travel distance…) of each VehicleGroup. |
| PriceGroupItem | PriceGroupItem | Describe detailed constraints for hourly rental options of each PriceGroup in system. |
| Garage | Garage | Describe all garages in system |
| GarageWorkingTime | GarageWorkingTime | Describe working time constraints of each Garage in system. |
| Location | Location | Describe all locations in system. |
| Nation | Nation | Describe all nations in system. |
| BookingReceipt | BookingReceipt | Describe all booking receipts in system. |

Table 9: Class dictionary

### Class Diagram Explanation

#### AspNetUser

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | String | Private | Unique identifier of AspNetUser. |
| Email | String | Private | Email of user. |
| EmailConfirmed | Boolean | Private | Status indicating whether email has been confirmed. |
| PasswordHash | String | Private | Encoded password of user. |
| SecurityStamp | String | Private | Stamp used to authenticate user’s cookies and session. |
| PhoneNumber | String | Private | Phone number of user. |
| PhoneNumberConfirmed | Boolean | Private | Status indicating whether phone number has been confirmed. |
| TwoFactorEnabled | Boolean | Private | Status indicating whether two-factor sign-in is enabled. |
| LockoutEndDateUtc | Nullable<DateTime> | Private | Date until which this user is locked out if lockoutEnable is true. |
| LockoutEnable | Boolean | Private | Status indicating whether lockout is enabled. |
| AccessFailedCount | Integer | Private | Number of failed access for this account. |
| UserName | String | Private | Username |
| FullName | String | Private | Full name of user |
| IsProviderUntil | Nullable<DateTime> | Private | Date until which this user has provider role. |
| AvatarURL | String | Private | URL of user’s avatar |

Table 10: AspNetUser class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 11: AspNetUser class – Methods

#### AspNetRole

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | String | Private | Unique identifier of AspNetRole |
| Name | String | Private | Name of role |

Table 12: AspNetRole class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 13: AspNetRole class - Methods

#### AspNetUserClaim

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| Id | Integer | Private | Unique identifier of claim |
| UserId | String | Private | ID of AspNetUser whom this claim belongs to |
| ClaimType | Stirng | Private | Type of claim |
| ClaimValue | String | Private | Value of claim |

Table 16: AspNetUserClaim class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 17: AspNetUserClaim class - Methods

#### AspNetUserLogin

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| LoginProvider | String | Private | Name of third-party login provider |
| ProviderKey | String | Private | Login key of user provided by third-party service |
| UserId | String | Private | ID of AspNetUser who can log in using this login |

Table 18: AspNetUserLogin class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 19: AspNetUserLogin class - Methods

#### Vehicle

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | String | Private | Unique identifier of vehicle |
| LicenseNumber | String | Private | License number of vehicle |
| Name | String | Private | Name of vehicle |
| ModelID | Integer | Private | Model of vehicle |
| Year | Integer | Private | Year in which this vehicle was produced |
| GarageID | Integer | Private | ID of Garage containing this vehicle |
| VehicleGroupID | Nullable<Integer> | Private | ID of VehicleGroup containing this vehicle |
| TransmissionType | Integer | Private | Transmission type of vehicle |
| TransmissionDetail | String | Private | Detailed description about transmisson of vehicle |
| FuelType | Nullable<Integer> | Private | Fuel type of vehicle |
| Engine | String | Private | Description about engine of vehicle |
| Color | Integer | Private | Color of vehicle |
| Description | String | Private | Description about vehicle |
| NumOfComment | Integer | Private | Number of customer’s feedback this vehicle has received |
| Star | Decimal | Private | Rating of vehicle |
| IsDeleted | Boolean | Private | Status indicating whether this vehicle was deleted |

Table 20: Vehicle class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 21: Vehicle class - Methods

#### VehicleImage

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | String | Private | Unique identifier of image returned by cloudinary |
| VehicleID | Integer | Private | ID of Vehicle which this image belongs to |
| URL | String | Private | URL of image |

Table 22: VehicleImage class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 23: VehicleImage class - Methods

#### VehicleModel

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of vehicle model |
| Name | String | Private | Name of role |
| BrandID | Integer | Private | ID of Brand of this model |
| NumOfDoor | Integer | Private | Number of door on this vehicle |
| NumOfSeat | Integer | Private | Number of seat in this vehicle |

Table 24: VehicleModel class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 25: VehicleModel class - Methods

#### VehicleBrand

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of vehicle brand |
| Name | String | Private | Name of brand |

Table 26: VehicleBrand class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: VehicleBrand class - Methods

#### Category

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of category |
| Name | String | Private | Name of category |

Table 26: VehicleBrand class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: VehicleBrand class - Methods

#### Garage

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of garage |
| OwnerID | String | Private | ID of AspNetUser owing this garage |
| Name | String | Private | Name of garage |
| LocationID | Integer | Private | ID of Location of garage |
| Address | String | Private | Address of garage |
| Email | String | Private | Email of garage |
| Phone1 | String | Private | Primary phone number of garage |
| Phone2 | String | Private | Secondary phone number of garage |
| NumOfComment | Integer | Private | Number of customer’s feedback this garage has received |
| Star | Decimal | Private | Rating of garage |
| Description | String | Private | Description of garage |
| Policy | String | Private | Policy of garage |
| IsActive | Boolean | Private | Status indicating whether this garage is active. |
| IsDeleted | Boolean | Private | Status indicating whether this garage was deleted. |

Table 26: Garage class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: Garage class - Methods

#### GarageWorkingTime

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| GarageID | Integer | Private | ID of Garage which this working time constraint is applied to. |
| DayOfWeek | Integer | Private | Day of week which this working time constraint represents. |
| OpenTimeInMinute | Integer | Private | Opening time of this garare in this day of week. |
| CloseTimeInMinute | Integer | Private | Closing time of this garage in this day of week. |

Table 26: GarageWorkingTime class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: GarageWorkingTime class - Methods

#### Location

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of location. |
| Name | String | Private | Name of location. |
| NationID | Integer | Private | ID of Nation to which this location belongs. |

Table 26: Location class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: Location class - Methods

#### Nation

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of nation |
| Name | String | Private | Name of nation |

Table 26: Nation class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: Nation class - Methods

#### VehicleGroup

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of vehicle group |
| OwnerID | String | Private | ID of AspNetUser whom this group belongs to |
| Name | String | Private | Name of group |
| IsActive | Boolean | Private | Status indicating whether this group is active |
| WithDriverPriceGroupID | Integer | Private | ID of default PriceGroup which is applied for driver-included rent. |

Table 26: VehicleGroup class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: VehicleGroup class - Methods

#### PriceGroup

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of price group. |
| DepositPercentage | Decimal | Private | Percentage of total rental price which customer has to pay to provider when booking. |
| PerDayPrice | Double | Private | Pricing of daily rental. |
| MaxRentalPeriod | Nullable<Integer> | Private | Maximum days a rental can last. |
| MaxDistancePerDay | Nullable<Integer> | Private | Maximum km that customer can travel in 1 days without being charged extra fee with the rented vehicle. |
| ExtraChargePerKm | Nullable<Double> | Private | Extra fee for every 1 km outside allowed travel distance. |

Table 26: PriceGroup class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: PriceGroup class - Methods

#### PriceGroupItem

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| PriceGroupID | Integer | Private | ID of PriceGroup to which this hourly rental option belongs. |
| MaxTime | Integer | Private | Rental duration of this rental option. |
| Price | Double | Private | Rental price of this rental option. |
| MaxDistance | Integer | Private | Maximum km that customer can travel without being charged extra fee with the rented vehicle using this rental option. |

Table 26: PriceGroupItem class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: PriceGroupItem class - Methods

#### BookingReceipt

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Integer | Private | Unique identifier of receipt. |
| CustomerID | String | Private | ID of AspNetUser who is the customer of this receipt. |
| RentalPrice | Double | Private | Total rental price. |
| Deposit | Double | Private | Deposit paid by customer to provider when booking. |
| BookingFee | Double | Private | Fee paid by customer to system when booking. |
| Distance | Nullable<Integer> | Private | Maximum distance customer can travel without being charged extra fee with the rented vehicle. |
| Star | Nullable<Integer> | Private | Rating from customer for this rental period. |
| Comment | String | Private | Feedback from customer about this rental reriod. |
| BookingTime | DateTime | Private | Time when this booking is made. |
| StartTime | DateTime | Private | Time when rental period starts. |
| EndTime | DateTime | Private | Time when rental period ends. |
| GarageID | Integer | Private | ID of Garage to which this booking belongs. |
| GarageName | String | Private | Name of Garage at booking moment. |
| GarageAddress | String | Private | Address of Garage at booking moment. |
| GaragePhone | String | Private | Phone1 of Garage at booking moment. |
| GarageEmail | String | Private | Email of Garage at booking moment. |
| VehicleID | Integer | Private | ID of booked Vehicle. |
| LicenseNumber | String | Private | License number of Vehicle at booking moment. |
| VehicleName | String | Private | Name of Vehicle at booking moment. |
| TransmissionDetail | String | Private | TransmissionDetail of Vehicle at booking moment. |
| Engine | String | Private | Engine of Vehicle at booking moment. |
| Color | Integer | Private | Color of Vehicle at booking moment. |
| IsCanceled | Boolean | Private | Status indicating whether this booking has been canceled. |
| IsPending | Boolean | Private | Status indicating whether this booking has not been paid. |

Table 26: BookingReceipt class - Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return type** | **Visibility** | **Description** |
| Getter | Attribute type | Public | Get value of attribute |
| Setter | Void | Public | Set value of attribute |

Table 27: BookingReceipt class - Methods

### Interactive Diagram

#### Web Application

##### Guest

##### Authenticated User

##### Manager

## User Interface Design

## Database Design

### Entity relationship diagram

Figure 127 : Entity relationship diagram

### Entity Dictionary

Table 11 : Entity dictionary

Table 12 : Detailed data dictionary

## Algorithms

### Recommender Engine

#### Introduction

As a car rental solution, CRP has to be able to introduce the most suitable vehicles to customers. This will helps improving not only the customers’ but also the providers’ satisfaction, and ultimately gaining more transactions, as well as interest for our application. Consequently, a recommender engine is necessary.

#### Common approaches

Two common approaches on designing recommender solution are *Collaborative filtering* and *Content-based filtering*.

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users’ behaviors, activities or preferences and predicting what users will like based on their similarity to other users. [[3]](#_References)

Content-based filtering methods are based on a description of the item and a profile of the user’s preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). [[4]](#_References)

Collaborative approach does not require analyzing the content of item that it recommend since it is based entirely on the user’s information. However, exactly because it depends solely on the data generated by users, it suffers the three common problem of computing, namely *cold* *start* (Lack of user’s interaction in the beginning), *scalability* (Scale badly when the number of user and their interaction increase), and *sparsity* (The number of item is too large compared to the number of user). Considering the low-number-of-transaction nature of car rental service when compares to other kind of services, scalability and sparsity issues can be evaded.

Content-based approach works well even under scarce user’s interaction environment since its recommendations are based on the items’ description. Its issue of not being able to recommend items with different content type (For instance, car and phone) is also not a problem, since CRP only has one type of item.

#### Solution’s approaches

CRP’s recommender design take a hybrid approach between content-based and collaborative.

Since our system’s recommended targets, namely vehicles, have many attributes that we can take advantage of (Brand, number of seat, color…), we initially tackles the problem using content-based methods. These includes *Vector space model* and *tf-idf.*

##### Represent items using vector space model

We first abstract the vehicles and their attributes by applying *Vector space model* [[5]](#_References), an item presentation algorithm.

Each vehicle is modeled as a vector (Refer to as *master vector* from now) in a multi-dimension space (Refer to as *vector space* from now), with each dimension corresponds to an attribute. If a vehicle has an attribute, the component vector (Refer to as *attribute vector* from now) corresponding to that attribute will has non-zero length.

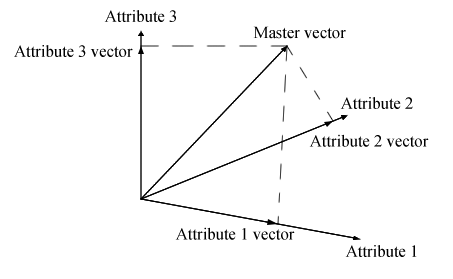


Figure 1 – Example for 3-dimensional vector space

There are several weighting scheme that can be used to calculate the attribute vector’s length. Amongst them, the most popular one is *tf-idf* (Term frequency – Inverse document frequency)[[6]](#_References), which. In our solution, tf-idf is applied with *binary scheme* (Further explanation will be given in *solution’s design* section).

##### Customer profile

A customer profile is another vector in the *vector space* which indicate a particular customer’s interest in vehicle, like which color or which fuel type that he has more affinity with.

In vector space model, the angle between 2 vectors determine the similarity between them. This means the smaller the angle between a vehicle’s *master vector* and a customer profile, the more similar that vehicle is to the customer’s reference.

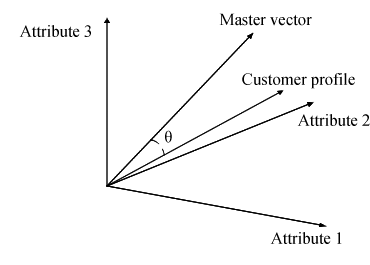


Figure 2 – Example for customer profile vector

It is common to use the cosine of this angle to represent their similarity, since their values are between 1 and -1. This form allows us to tell at how many percent a customer will like a vehicle (Positive value), or dislike it (Negative value). Our recommender engine will find the cosine between each vehicle’s *master vector* and a customer profile, then recommend vehicles with the highest cosine value to that customer.

##### Mix in collaborative element

To further improve the diversity of our solution’s recommendation, we apply collaborative methods into our content-based solution. Our collaborative approach work under the assumption that other customers that has booked the same vehicle with this customer (Refer to as neighbor from now) will have similar vehicle reference; and the more *neighbors* a vehicle has, the more similar it is to this customer’s reference.

Our collaborative approach introduces new attributes into the *vector space*. Each of these new attribute represent a neighbor and whether the item has been booked by this neighbor before.

|  | Neighbor 1 | Neighbor 2 | Neighbor 3 | Neighbor 4 |
| --- | --- | --- | --- | --- |
| Vehicle 1 | Yes | Yes | No | No |
| Vehicle 2 | No | Yes | No | No |
| Vehicle 3 | Yes | Yes | Yes | No |

Table 1 – Example of vehicle with 4 neighbor attributes

#### Solution’s design

##### How to calculate cosine of angle between 2 vectors

Cosine of the angle θ between 2 vectors and in n-dimension *vector space* can be calculated as follow:

where is dot product of 2 vectors, and is the norm of each vector.

The dot product can be calculated as follow:

The norm of a vector can be calculated as follow:

##### Apply tf-idf weighting scheme

Applying tf-idf, length, or ‘weight’ of each *attribute vector* of can be calculated as follow:

idf (Inverse document frequency) can be calculated as follow:

with being the total number of item and being the total number of item that has attribute i.

The scheme to calculate tf will be discussed in the next section.

##### Binary representation of attributes

In our approach, raw attributes can only either appear or does not appear in an item. This leads to representing them as binary values. Under this form, we can apply *binary tf weighting scheme* of *tf-idf*, where tf weight equals the raw binary value.

|  | 4-seat | 7-seat | Gasoline | Diesel | Neighbor 1 | Neighbor 2 |
| --- | --- | --- | --- | --- | --- | --- |
| Vehicle 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Vehicle 2 | 1 | 0 | 0 | 1 | 0 | 1 |
| Vehicle 3 | 0 | 1 | 1 | 0 | 1 | 1 |
| Vehicle 4 | 0 | 1 | 1 | 0 | 1 | 0 |
| Vehicle 5 | 1 | 0 | 0 | 1 | 0 | 0 |

Table 2 – Example tf values of vehicles with 6 binary attributes

With this approach, there is also no need to apply normalization to eliminate item’s size bias (For document-like item, the total number of ‘word’ in them varies, meaning bigger document will more likely to have more ‘hit’(tf) for each search term/attribute. However, with binary attributes, an item either ‘has’ or ‘does not have’ an attribute. In other words, tf is limited to 1 and 0, and therefore has no such bias).

| **attribute** | | 4-seat | 7-seat | Gasoline | Diesel | Neighbor 1 | Neighbor 2 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **tf** | Vehicle 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Vehicle 2 | 1 | 0 | 0 | 1 | 0 | 1 |
| Vehicle 3 | 0 | 1 | 1 | 0 | 1 | 1 |
| Vehicle 4 | 0 | 1 | 1 | 0 | 1 | 0 |
| Vehicle 5 | 1 | 0 | 0 | 1 | 0 | 0 |
| **df** | | 3 | 2 | 3 | 2 | 3 | 3 |
| **idf (With D = 5)** | | 0.097 | 0.222 | 0.097 | 0.222 | 0.097 | 0.097 |
| **weight** | Vehicle 1 | 0.097 | 0 | 0.097 | 0 | 0.097 | 0.097 |
| Vehicle 2 | 0.097 | 0 | 0 | 0.222 | 0 | 0.097 |
| Vehicle 3 | 0 | 0.222 | 0.097 | 0 | 0.097 | 0.097 |
| Vehicle 4 | 0 | 0.222 | 0.097 | 0 | 0.097 | 0 |
| Vehicle 5 | 0.097 | 0 | 0 | 0.222 | 0 | 0 |

Table 3 – Example weight of 5 vehicles with 6 binary attributes

##### Build the customer profile

We can build a user profile by using that user’s booking history as reference. Every booking has all the necessary data to construct an item vector similar to vehicle vector.

| # | Vehicle | 4-seat | 7-seat | Neighbor 1 | Neighbor 2 | Star |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Vehicle 1 | 1 | 0 | 1 | 1 | 2 |
| 2 | Vehicle 5 | 1 | 0 | 0 | 0 | 1 |
| 3 | Vehicle 3 | 0 | 1 | 1 | 1 | 4 |
| 4 | Vehicle 2 | 1 | 0 | 0 | 1 | 5 |
| 5 | Vehicle 3 | 0 | 1 | 1 | 1 | - |

Table 4 – Example of 5 bookings with 4 attributes

The bookings also have star-rating, which we can utilize to determine whether the customer liked or dislike the booking. In our approach, we assume that a rating lower than 3-star indicates *dislike* (-1), higher than 3-star indicates *like* (1) and equals 3 or empty rating means *neutral* (0).

| # | Vehicle | 4-seat | 7-seat | Neighbor 1 | Neighbor 2 | Star | Like |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Vehicle 1 | -1 | 0 | -1 | -1 | 2 | -1 |
| 2 | Vehicle 5 | 1 | 0 | 0 | 0 | 1 | -1 |
| 3 | Vehicle 3 | 0 | 1 | 1 | 1 | 4 | 1 |
| 4 | Vehicle 2 | 1 | 0 | 0 | 1 | 5 | 1 |
| 5 | Vehicle 3 | 0 | 0 | 0 | 0 | - | 0 |

Table 5 – Example tf of 5 bookings with 4 attributes applying like-dislike scheme

tf of each of the customer profile’s *attribute vector* can be calculated as tf of sum of tf of every booking’s attribute vector of the same dimension. Since these sum are no longer binary, we calculate customer profile’s of dimension i using *log normalization* scheme instead:

In this equation, m is the total of booking. This equation has been modified from the original scheme to accomplice negative value. The customer profile’s vector can be calculated using our default tf-idf weighting scheme.

| **Attribute** | # | Vehicle | 4-seat | 7-seat | Neighbor 1 | Neighbor 2 |
| --- | --- | --- | --- | --- | --- | --- |
| **Booking’s tf** | 1 | Vehicle 1 | -1 | 0 | -1 | -1 |
| 2 | Vehicle 5 | -1 | 0 | 0 | 0 |
| 3 | Vehicle 3 | 0 | 1 | 1 | 1 |
| 4 | Vehicle 2 | 1 | 0 | 0 | 1 |
| 5 | Vehicle 3 | 0 | 0 | 0 | 0 |
| **df** | | | 3 | 1 | 2 | 3 |
| **idf (D=5)** | | | 0.097 | 0.398 | 0.222 | 0.097 |
| **Customer profile** | **tf** | | -1 | 1 | 0 | 1 |
|  | **weight** | | -0.097 | 0.398 | 0 | 0.097 |

Table 6 – Example weight for attribute vectors of a customer profile

##### Calculate vehicle’s score

As mentioned, the score used to recommend vehicle will be the cosine between the vehicle master vector and the customer profile.

| Attribute | | | 4-seat | 7-seat | Neighbor 1 | Neighbor 2 |
| --- | --- | --- | --- | --- | --- | --- |
| **Customer profile** | | | -0.097 | 0.398 | 0 | 0.097 |
| **Vehicle** | **tf** | Vehicle 1 | 1 | 0 | 1 | 1 |
| Vehicle 2 | 1 | 0 | 0 | 1 |
| Vehicle 3 | 0 | 1 | 1 | 1 |
| Vehicle 4 | 0 | 1 | 1 | 0 |
| Vehicle 5 | 1 | 0 | 0 | 0 |
| **df** | | 3 | 2 | 3 | 3 |
| **idf (D=5)** | | 0.097 | 0.222 | 0.097 | 0.097 |
| **weight** | Vehicle 1 | 0.097 | 0 | 0.097 | 0.097 |
| Vehicle 2 | 0.097 | 0 | 0 | 0.097 |
| Vehicle 3 | 0 | 0.222 | 0.097 | 0.097 |
| Vehicle 4 | 0 | 0.222 | 0.097 | 0 |
| Vehicle 5 | 0.097 | 0 | 0 | 0 |
| **Score** | Vehicle 1 | 0 | | | |
| Vehicle 2 | 0 | | | |
| Vehicle 3 | 0.889907 | | | |
| Vehicle 4 | 0.866331 | | | |
| Vehicle 5 | -0.23042 | | | |

Table 7 – Example score for 5 vehicles with 4 attributes

A positive value shows us the probability the customer will like the vehicle, while a negative one shows us the probability the customer will dislike the vehicle.

#### Algorithm’s time complexity

Consider a *vector space* with n attributes, k vehicles, a customer with m bookings in her booking history, and Math.Log10(double) and Math.Sqrt(double) has O(1) time complexity, we can calculate the time complexity of this recommender algorithm.

##### Build customer profile

| **Step** | **Complexity** |
| --- | --- |
| Calculate each attribute vector of the profile | O(m) |
| * Calculate | O(m) |
| * Calculate | O(m) |
| * Calculate | O(1) |
| * Calculate | O(m) |
| * Calculate | O(m) |
| * Calculate | O(1) |
| * Calculate | O(1) |
| **Total** | **O(nm)** |

Table 8 – Time complexity of **Build customer profile** step

##### Build vehicle vectors

| **Step** | **Complexity** |
| --- | --- |
| Calculate each vehicle vector | O(nk) |
| * Calculate each attribute vector of | O(k) |
| * Calculate | O(k) |
| Calculate | O(k) |
| Calculate | O(1) |
| * Calculate | O(1) |
| **Total** | **O(nk2)** |

Table 9 – Time complexity of **Build vehicle vectors** step

##### Score vehicles

| **Step** | **Complexity** |
| --- | --- |
| Calculate norm | O(n) |
| Calculate each vehicle’s score | O(n) |
| * Calculate norm | O(n) |
| * Calculate dot product | O(n) |
| * Calculate vehicle’s score | O(1) |
| **Total** | **O(nk)** |

Table 10 – Time complexity of **Score vehicles** step

##### Overall time complexity

The overall time complexity of our algorithm is **O(n(k2 + k + m))**. This complexity has 2 pain points. The first being k2, which can somehow be reduced by applying normal filtering first to lessen the number of vehicle in vector space. The second pain point is the number of collaborative attributes in n.

n is the sum between the content-based attributes presented naturally on every vehicle and the collaborative attributes which are the neighbors a vehicle has. The more booking the customer makes, the more neighbors he may have, and the bigger n will become. However, as a car rental solution, our transactions tend to be big in value but small in quantity. The average number of booking a single customer makes may just stop at a few dozen. Further conclusion can only be drawn with concrete statistics, however.

Conclusively, this algorithm’s scalability is quite poor, but this is common and is still very acceptable as a recommendation solution, especially those follow collaborative approach. In the future that the system will increase in size, there will be a need for either an upgrade of hardware and software to increase computing power, or the development of a new engine.

# システムの展開とテスト

## はじめに

### 概要

本節、物理データベースの詳細とその記述は表れています。その上、BMAのテストの方法やテストプランやテストケースも記載されています。

### テストアプローチ

* ゴール: 要求を満たすため、すべてのシステムが正常に実行されていること。
* 方法: システムテスト, ブラックボックステスト。

## データベース関係ダイアグラム

### ウェブアプリケーション物理的なダイアグラム

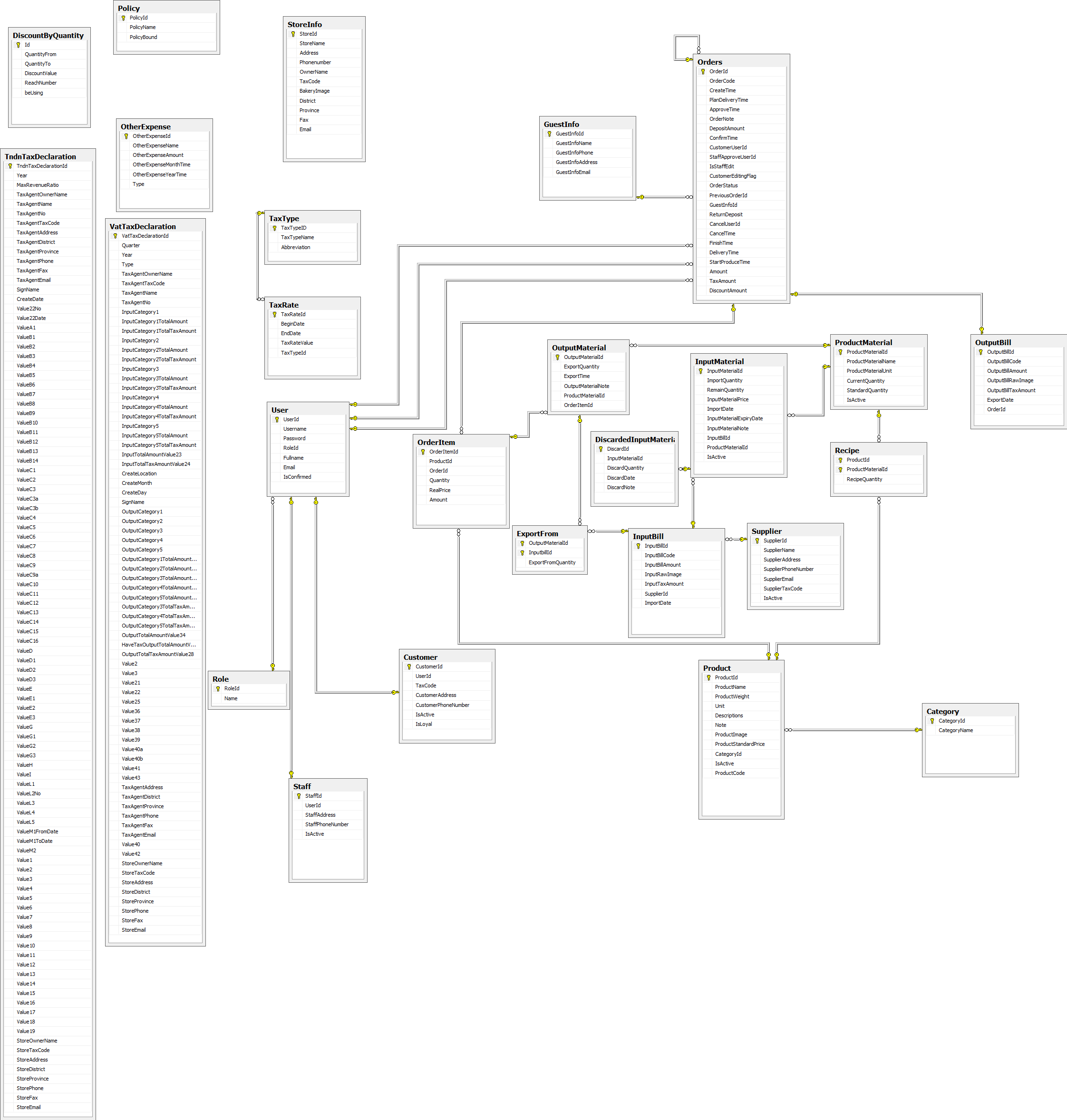


Figure 129: ウェブサイトアプリケーション物理的なダイアグラム

### 2.2 ウェブアプリケーションデータ辞書

## パフォーマンス対策

パフォーマンスを測定するために、デプロイとともにazure ホスティングでテストして、予測を行いました。

### 製品検索、メンバーのパフォーマンス

一般的に、結果はユーザが検索フォームを提出した後、5秒未満を現されます。

### 送り状リストが展示されるパフォーマンス

一般的には、ユーザーが「送り状リスト」のリンクをクリックすると、注文リストはより5秒以内に展示されます。

### 収入統計など展示されるパフォーマンス

一般的に、時間を選んで、「統計」というボタンをクリックすると、10秒以内で収入統計など展示されます。

## テスト計画

### テストする機能

システムテストはコアワークフローに基づいて行うことになります。

テストする機能:

* ユーザログイン
* ゲスト：
  + 製品を注文します
* 顧客：
  + 製品を注文します
  + 送り状を確認します
* 人員
  + 送り状を造ります
  + 送り状を承認します
  + 送り状の状態を変化します
  + 送り状を編集します
  + 入力原料を付加します
  + 入力原料を捨てます
  + 入力法案を付加します
* 管理
  + 収入統計
  + 製品別収益
  + お客様による純益統計
  + 消費税申告
  + 免許税申告

### テストされない機能

* ログ‐アウト
* プロフィールを編集
* 消費税申告
* 免許税申告
* 顧客管理
* 人員管理

## システムテストのテストケース

# ソフトウェアユーザーズマニュアル

## インストレーションガイド

### サーパ側の環境を設定

#### ハードウェア要件

**スムーズにウェブサイトを実行するてめには、以下の要件を備えたコンピュータが必要になります:**

* オペレーティングシステム: ウィンドウXPまたはウィンドウのそれ以降のバージョン。
* プロセッサ：１GHｚ以上。
* メモリ：2GB以上。
* ハードウェア：2GB以上のハードドライブスペース。
* Wifi インターネット(> 2Mbps)。

#### ソフトウェア要件

* + - ウェブサーバー ：Internet information system – IIS 8 With MVC5 enable
    - Microsoft Windows Server 2008 R2 Enterprise (Server Pack 1 64 bit): オペレイティングシ
    - .Net Framework 4.5.1
    - SQL Server 2008 R2

### サーパ側で展開

#### 展開パッケージの準備

* データベース　ファイル： BMA.mdf
* サーパーでのフォルダにソースコードファイル（BMA）をコピーする。
  + 例えば：C:\BMA

#### サーパーの構成前に展開

* **データベースの構成：**

<connectionString>

<add name = “DefaultConnection”

connectionString =”Datasource:(local);Initial Catalog=BMA;Persist Security Info = True;user id=sa;password=123456789” providerName = “System.Data.SqlClient” />

<add name = “BMAChangeDB”

connectionString =”Datasource:(local);Initial Catalog=BMA;Persist Security Info = True;user id=sa;password=123456789” providerName = “System.Data.SqlClient” />

</connectionString>

* Catalog : データベース名。
* User id :ユーザ名データベース。
* Password :パスワード

#### サーパ側でWebアプリケーションの展開

* **データベース展開：**
* ＳＱＬ Server 2008 R2 にBMA.mdfをアタックしてください。
* **ソースコードの展開：**
* オーペンIISマネージャ
* “Default Web Site”にマウスを右クリックします。“Add Application”を選びます。
  + Alias : Aliasを入力してください。例えば：ＢMA
  + Application pool : ASP.NET v4.0 を選んでください
  + Physical path:サーパーのソースコードファイルを選んでください（C:\BMA）
* “OK”ボタンをクリックしてください。
* オーペン：<http://localhost/BMA>

### クライアント側の環境の設定

#### コンピュータのための設定

* クライアントデバイスは、ウェブサイトを起動するには、以下のブラウザのいずれかを持っている必要があります：
  + Google Chrome(ダウンロードリンク):
    - <http://www.google.com/Chrome>
  + Coc Coc（コックコック）：
    - <http://www.coccoc.com/>

## ユーザーガイド

### ログイン



図面140: ログインポップアップ

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 口座を入力して |
| **2** | パスワードを入力して |
| **3** | 「Đăng nhập」ボタンをリックして |
| **4** | パスワードを忘れたら、「Quên mật khẩu」ボタンをリックして |

テーブル13: ログインポップアップ

### 製品を注文すること



図面 141:製品を注文すること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 製品を探す |
| **2** | 「Mua hàng」 ボタンをクリックして、 製品を注文する |

テーブル 14:　製品を注文する

### カートに製品を追加すること



図面 142: カートに製品を追加すること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | この製品の数量 |
| **2** | 「Thêm vào giỏ hàng 」 ボタンをくりっくして、カートに製品を追加する |

テーブル 15: カートに製品を追加する

### かーとをチェックすること



図面 143: かーとをチェックすること

|  |  |
| --- | --- |
| Step | 説明 |
| **1** | 「Giỏ hàng 」 ボタン をクリックして、製品のリストが見える |
| **2** | 「Thanh toán 」 ボタン をクリックして、カートの精細を見える |

テーブル 16: かーとをチェックすること

### カーとの精細



図面 144: カートの精細

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 製品の数量を変われる |
| **2** | 製品のリストを戻って、製品を注文して行く |
| **3** | 「Cập nhật giỏ hàng 」 ボタンをクリックして、 カートを更新する |
| **4** | 配送日をセレクトする |
| **5** | 「Đặt hàng 」ボタン をクリックして、注文を完成する |

テーブル 17: カートの精細

### 送り状を確認すること



図面 145: 送り状を確認すること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | カートの精細を戻れば、「Quay lại giỏ hàng 」 ボタン をクリックする |
| **2** | 送り状の確認のボタン |

テーブル 18: 送り状を確認すること

### 送り状に顧客の情報を充実すること



図面 146: 送り状に顧客の情報を充実すること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | これに成員をログインする |
| **2** | 成員なければ、情報を充実するところ |
| **3** | 「Quay lại giỏ hàng 」 ボタン をクリックして、送り状の確認ページを戻る |
| **4** | 口座とパスワードを入力して,「Đăng nhập 」ボタンをクリックして、送り状を完成する |
| **5** | 顧客の情報を入力して,「Hoàn tất 」 ボタンをクリックして、送り状を完成する |

テーブル 19: 送り状に顧客の情報を充実すること

### 送り状のリストを見ること



図面 147: 送り状のリストを見ること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 不足の原料のリスト |
| **2** | 新し送り状を造る |
| **3** | 送り状の精細 |
| **4** | 送り状を変わる |
| **5** | 送り状を取り消す |
| **6** | 探す |

テーブル 20: 送り状のリストを見ること

### 「人員」送り状を造ること



図面 148: 「人員」送り状を造ること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 製品 の単価 |
| **2** | 製品の数量 |
| **3** | 送り状に製品をテーキオフする |
| **4** | 送り状をキャンセルこと |
| **5** | 送り状を続く |
| **6** | 送り状に製品を追加する |

テーブル 21: 「人員」送り状を造ること

### 送り状に顧客を追加すること



図面 149: 送り状に顧客を追加すること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 顧客をセレクトする |
| **2** | 顧客なければ、口座を造る |
| **3** | お客さんの名前 |
| **4** | お客さんの口座 |
| **5** | お客さんのメール |
| **6** | お客さんの住所 |
| **7** | お客さんの電話番号 |
| **8** | お客さんの税額 |
| **9** | 入力をすべてリセット |
| **10** | 口座を造って、この口座にその送り状を追加する |
| **11** | 送り状を造るのページを戻る |
| **12** | 送り状をキャンセルする |

テーブル 22: 送り状に顧客を追加すること

### 新い顧客の口座で送り状をチェックアウト



図面 150: 新顧客の口座で送り状をチェックアウト

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 新顧客の情報 |
| **2** | 送り状の情報 |
| **3** | この送り状に原料数量 を過ごす |
| **4** | 送り状にお客さんを追加する ページを戻る |
| **5** | 送り状を造るのページを戻る |
| **6** | この送り状を確認する |
| **7** | この送り状をキャンセルする |

テーブル 23: 新顧客の口座で送り状をチェックアウト

### 入力法案 リスク



図面 151: 入力法案 リスク

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 新入力法案を造る |
| **2** | 探す |
| **3** | 入力法案 リスク |
| **4** | 入力法案 の精細を見る |
| **5** | 入力法案 の情報を変わる |

テーブル 24:入力法案 リスク

### 新入力法案を造ること



図面 152: 新入力法案を造ること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | サプライヤー をセレクトする |
| **2** | この入力法案料(VATがない) |
| **3** | VAT の料金 |
| **4** | 輸入日 |
| **5** | 入力法案の写真 |
| **6** | 入力法案 陸とを戻る |
| **7** | 新入力法案を造る |

テーブル 25: 新入力法案を造ること

### 入力原料 リスクを見る



図面 153: 入力原料 リスク

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 新入力原料を造ること |
| **2** | 探す |
| **3** | 入力原料 リスク |
| **4** | 入力原料 精細 |
| **5** | 入力原料 の情報を変わる |
| **6** | 入力原料をデリーとする |

テーブル 26: 入力原料リスク

### 入力原料 の精細を見ること



図面 154:入力原料 の精細を見ること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 入力原料の情報 |
| **2** | 入力原料をデリートする |
| **3** | 同じ入力法案の入力原料リスク |
| **4** | 入力原料の脚注 |
| **5** | 入力原料デリーとの情報 |
| **6** | 入力原料 リスクを戻る |
| **7** | 入力原料 の情報を更新する |

テーブル 27:入力原料の精細を見ること

### 新入力原料を造ること



図面 155: 新入力原料を造ること

|  |  |
| --- | --- |
| ステップ | 説明 |
| **1** | 入力原料名 |
| **2** | 入力原料の数量 |
| **3** | 入力原料の総価格 |
| **4** | 入力原料の輸入日 |
| **5** | 入力原料の有効期限 |
| **6** | 入力原料の脚注 |
| **7** | 入力原料の 入力法案 のコード |
| **8** | 入力原料 リスクを戻る |
| **9** | 入力原料を造る |

テーブル 28:入力原料を造ること

# Appendix

1. Sashimi – Modified Waterfall Model:

<http://www.waterfall-model.com/sashimi-waterfall-model>

1. UML 2.0 standard

<http://www.omg.org/spec/UML/2.0>

1. Collaborative filtering approach

[https://en.wikipedia.org/wiki/Recommender\_system#Collaborative\_filtering](https://en.wikipedia.org/wiki/Recommender_system%23Collaborative_filtering)

1. Content-based approach

[https://en.wikipedia.org/wiki/Recommender\_system#Content-based\_filtering](https://en.wikipedia.org/wiki/Recommender_system%23Content-based_filtering)

1. Vector space model

<https://en.wikipedia.org/wiki/Vector_space_model>

1. tf-idf

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>