**Capstone Project Document template**

***A. Table of contents for the capstone project document***

Students must include a table of contents with their results report. The following is a template, provided for reference.

* + Front page (see next section)
  + Table of contents
  + Index [Optional]
  + Report No1: Introduction
  + Report No.2: Software Project Management Plan (SPMP)
  + Report No.3: Software Requirements Specification (SRS)
  + Report No.4: Software Design Description (SDD)
  + Report No.5: Software Test Documentation (STD)
  + Report No.6: Software User’s Manual (SUM)
  + Appendix:

## . Glossary [Optional]

## <Place all definitions or abbreviation used in this document >

## . References [Optional]

## <Place all referenced materials used in this document >

## . Others [Optional]

***B. Front page template***

Next page is the template of front page of CapStone project document. Groups must follow this template. In this template: Replace any text in <> with your text. When you are done, there should be no <> or text surrounded by <> in this front page. Remember, the purpose of this page acts as a cover; it will not be graded.

|  |  |
| --- | --- |
| **Logo_FPT_University_doc** | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Long-distance bus reservation |
|  |
| |  |  | | --- | --- | | **Group 4** | | | **Group Members** | Nguyen Son Truong – Team Leader – 60466  Nguyen Luong Hai – Member – 60…  Nguyen Thi Bich Tram – Member – 60…  Nguyen Ngoc Son – Member – 60… | | **Supervisor** | Kieu Trong Khanh | | **Ext Supervisor** | - | | **Capstone Project code** | Bus Reservation | |
|  |

|  |
| --- |
| - Ho Chi Minh City, 01/2013 - |

*<The Table of Contents goes here>*

# Introduction

Every year in holidays, travel needs are rising, many people move out the city for relaxing or home town visiting. Especially in big cities like Hanoi or Ho Chi Minh City, it is always overloaded for the ticket booking system in all the ticket offices. Customers have to wait for hours to have a chance to buy a ticket.

* 1. **Current travelling service system**
     1. Vehicle managing
  + The buses are grouped into teams by its route, and each team has a fixed route and terminations.
  + Each team has a Team Manager (TM), who has all the status of their bus in their group. The TM will control the schedule of the buses and manage the seat in each bus. They report daily to the Operator Officers (OO) about the buses status and schedule.
  + The status of the buses help the TM controls each bus on their team and builds the schedule base on it. The status is defined by following :
    - Available: the bus is not planned for any departure date.
    - Waiting: the bus is in the station and waiting for departure date.
    - Busy: the bus is on their trip.
    - Maintenance: the bus is on maintaining.
  + Each bus will have a routine of departure date.
  + If the route is always in full or lacking seat, TM will make a request to the OO to have more bus to their team.
  + Otherwise, if the route usually remains excess seats, the TM will request for returning some buses back to OO and extended the time between departure dates.
  + And if excess seats problem occurs in a long time, the TM may ask for removing the routes and all buses in the team will return to OO.
    1. Ticket managing
  + The entire scheduled journey will be submitted to OO by the TM. Each journey will have a fitted fare defined by the policies of the company which are changed frequently by the OO.
  + Once schedule of the bus and seats division are planned, TM will send them to OO. The OO will base on ticket fare table/checklist to calculating the fare for each ticket, and then they will be sent to ticket offices to be sold.
  + If the sold tickets less than the minimum of the planned tickets, then the route will be canceled and customer will be return all the tickets fee and some bonus, or they can request for the other ticket in the next bus.
  + Customers can purchase for a ticket directly at the ticket offices or they can book tickets through the phone. Telephonist or staffs at the ticket offices will note down customers’ information and booking detail.
  1. **Problem Definition**
* On holidays, limited number of box offices makes it difficult to serve a large number of customers at a time
* It is a hard job for OO to manage all the bus status, route, fare and ticket balancing.
* It hard to control directly bus-to-bus when the bus status are not updated permanently to OO. All the bus status information only can be access through the TM.
* The number of tickets for each type (full journey and constituents of a journey) is fixed which make it inflexible. There’re cases where it lacking of tickets for a segment whiles others types of tickets are excess.
* When customer need to return or change tickets (change type of tickets, departure date, seats), only a small request would be approved.
  1. **Proposed Solution**

A system needs to be established which takes care of booking system, journey scheduling and managing vehicles to reduce human effort on operating and managing.  
The system will be hosted on the web, which provides easy access for customer to use the booking services. Customer can have their own account for viewing their transaction history. Scheduling and managing services are available for authorized staffs only.  
In detail, the system will enable the staffs and customers to do the following:

* + 1. **Non-Financial**
* Customer will be able to search by departure/arrival, date of departure, vehicle type... and view all available journey on the system, which is scheduled by the staffs through a scheduling service.
* Customer can easily registered for an account in the system. This account will maintain the history of the transaction they made, their information including full name, address, and phone number and so on.
* The system will allow each customer to contact one another and change their seat on a journey.
* The system will provide a method to manage all of the vehicles and their status to make it easier to track and make the schedule.
  + 1. **Financial**
* Payment for each booking transaction has to be done by the system and credited to the customer account. The system should be able to provide the customer a method of online payment like PayPal.
* A refund method should be provided for the customers in case they want to cancel and return their tickets or the journey is cancelled by the company.
* Fare and promotion plans is also managed by the system according to the policies of the company.
  1. **Functional Requirements**

Functional requirements of the proposed system are listed as below:

* + 1. **Search for journey**
* Users search by departure/arrival, date, coach type and number of passengers
* Display search result
  + Information of the journey: date-time, number of remaining seats, price
  + Information is ordered by type of journey (full-journey is displayed first, then its constituents.)
    1. **Booking**
* Users select journey based on search results
* Users select seats
  + Display seat map for corresponding coach type
  + User can select position on map
  + Registered users can switch seat position for others
* Payment using PayPal
* Output ticket code
  + User can view journey info by searching using ticket code
* Cancel and refund
  + 1. **Scheduling**
* Create route and schedule
* Update route/schedule
* Automatically cancel a journey if number of passengers less than 25% of seat numbers
  + Inform and refund for customers if the journey is cancelled
    1. **Vehicles Management**
* Add new coach
* Update coach information and status
* Coach category management
  + 1. **Fare and Promotions**
* Fare is based on route, coach type and departure time (for example: price on holidays is higher than usual)
* Promotion for customer loyalty or special events
  + 1. **User management**
* Registration of members (customers) to the site
* View transaction history
* View user info and notifications
* Update user info and status

# Software Project Management Plan (SPMP)

* 1. **Project** **Overview**

The name of this project is “Long-distance bus reservation” or “Bus Reservation”. It is a website for bus operators to manage their vehicles, build up schedule and journeys, and provide their customer a fastest way to get tickets on-line.

* + 1. **Problem Abstract**

As described in section 1, the needs of travelling is raising these days. Based on statistics from Mien Dong Bus Station of Ho Chi Minh City website ([http://www.benxemiendong.com.vn/](http://www.benxemiendong.com.vn/article/qua-trinh-phat-trien-14.html) ), in 2010:

* Average number of bus departures in a day is about 1,124. On holidays it can raise to 2,000.
* Average number of passengers in a day is 23,000 and 62,000 on holidays.

With a large amount of passengers and journeys in a day like that, apparently bus operators will be overloaded by setting schedule, managing their vehicles and selling tickets. Their customers will be frustrated by having to wait for hours to buy a ticket to get home.

* + 1. **About the System**
       1. **Boundaries of the system**

The system provides utilities for both bus operators and their customers. There will be a site for customers to book for their trips and pay for the tickets online, the customers may have their own accounts which help them access to transaction history and modify their information. There will also be a separated site for operators’ staff to managing the vehicles, scheduling for new trips or cancel trips.

Detailed functional requirements are mentioned in section 1.4

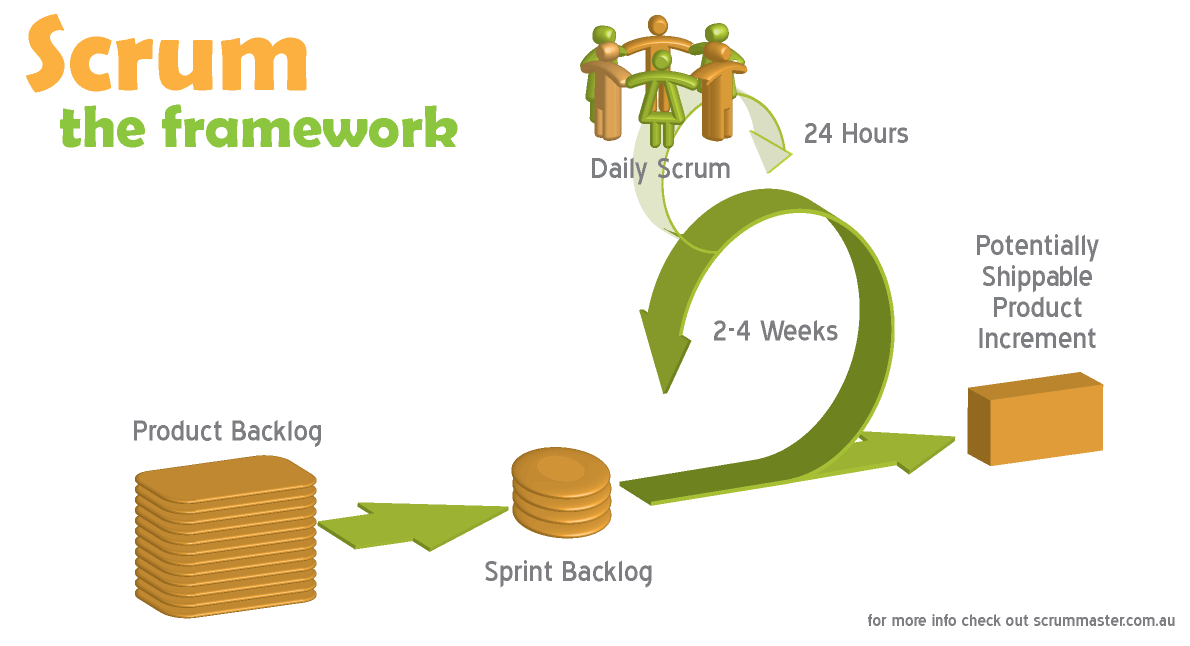
* + - 1. **Development Environment**

Below is the list of hardware and software requirements needed for development

**Hardware requirements:**

* Personal computer for developing with minimum configuration of 1GB RAM and 20GB of hard disk, Dual Core 2.0 GHz.
* Server computer for testing with minimum configuration of 2GB RAM and 20GB of hard disk, Dual Core 2.0 GHz.

**Software Requirements:**

* Operating system: Windows 7, Windows Server 2008
* IDE: Eclipse Indigo.
* DBMS: MySQL Server 5.5
* Source control: Tortoise SVN.
  1. **Project organization**
     1. **** **Software Process Model**

The model used for developing this project is Scrum model.

Figure 2.2.1. Scrum Model

Image source: http://www.scrummaster.com.au

* + 1. **Roles and Responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Full name | Role | Responsibilities |
| 1 | Kieu Trong Khanh | Project Owner | * Specify user requirements * Control the development process * Support in technique and business analysis |
| 2 | Nguyen Son Truong | Scrum Master  Developer  Tester | * Manage process * Design database * Develop and analyze requirement * Prepare documents * Code * Test |
| 3 | Nguyen Luong Hai | Developer  Tester | * Design database * Develop and analyze requirement * Build framework * Support technique * Code * Test |
| 4 | Nguyen Thi Bich Tram | Developer  Tester | * Develop and analyze requirement * Prepare documents * Create test plan and system test cases * Code * Test |
| 5 | Nguyen Ngoc Son | Developer  Tester | * Develop and analyze requirement * Create prototype and design GUI * Code * Test |

* + 1. **Tools and Techniques**
* Front-end: HTML, CSS, jQuery, AJAX
* Server: Apache Tomcat
* Framework: Struts 2, Spring 3, Hibernate 3

* 1. **Project management plan**
* This project will finished in 7 sprints, each sprint lasts for 2 weeks.
* Daily scrum meeting is held every day at 22:00 via Skype
* Sprint review meeting is at 8:30 on last Saturday of every sprint. Plan for the next sprint is also created in this meeting.
  + 1. **Product backlog**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Story ID** | **Story name** | **Acceptance Criteria** | **Task Id** | **Task description** | **Sprint** | **Owner** | **Status** |
| 1 | Create project introduction document | Introduction of overall project is recorded. | 1.1 | Research information about the current system | 1 | Tram | Done |
| 1.2 | Analyze the problem of the current system, and propose a solution | 1 | Truong | Done |
| 2 | Create task list | Product backlog is recorded. | 1.3 | Identify main requirements of the system and create task list base on them | 1 | Truong | Done |
| 3 | Build framework | Framework is ready for developing. | 1.4 | Build framework: Struts 2, Spring, Hibernate | 1 | Hai | Done |
| 4 | Create prototype | Master page is created. | 1.5 | Create UI for master page and search page | 1 | Son | Done |
| 5 | Create Software requirement specification | All the requirements are cleared and recorded. | 2.1 | Analyze and clarify requirements and create the document. | 2 | Team |  |
| 6 | Design Database | Database is created with sample data | 2.2 | Create entity-relationship diagram | 2 | Truong | Done |
| 2.3 | Design and create database based on entity-relationship diagram | 2 | Hai | Done |
| 2.4 | Test database | 2 | Team |  |
| 7 | As a user, I can search for a one-way trip. | Can I search for a one-way trip providing departure, arrival information, number of passenger and preferred bus type? | 3.1 | Create detail design for search function | 3 | Truong |  |
| 3.2 | Code search function | 3 | Truong |  |
| 4.1 | Create test case for search function | 4 | Truong |  |
| 4.2 | Test search function | 4 | Truong |  |

2.3.2. Task Sheet: Assignments and Timetable

2.3.3. All Meeting Minutes

* 1. **Coding Convention**
     1. **Naming Convention**

• Private Fields: underscore followed by lowerCamelCase. (Example: int \_privateField)

• Non-private Fields and properties: UpperCamelCase (or PascalCase). (Example: int PublicField)

• Local variables: lowerCamelCase. (Example: int localVariable)

• Do not use consecutive underscores in name.

• Do not use Hungarian style.

* + 1. **Lengths**

• Public name should not be longer than 32 characters or 7 words.

• Methods should contain no more than 70 lines of code (if it is, it must be divided into methods).

• Methods should contain no more than 5 levels of indentation (if it is, it must be divided into methods).

• A line of code should contain no more than 80 characters.

**2.5. Other material (if any)**

# Software Requirements Specifications (SRS)

* 1. **User Requirement Specification**

*<Summarize the customer requirements in a compact form>*

* 1. **System Requirement Specification (Specific Requirements)**

*<Summarize the system requirements in a compact form>*

3.2.1. External Interface Requirements

3.2.1.1. User Interfaces

3.2.1.2. Hardware Interfaces

3.2.1.3. Software Interfaces

3.2.1.4. Communications Protocol

3.2.2. System Features



Figure 3‑1 System use case

3.2.2.1. Search for trips

3.2.2.1. Search for trips use case



Figure 3‑2 Search for trips use case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE-UC001 SPECIFICATION** | | | | |
| **Use-case No.** | UC001 | **Use-case Version** | | v1.0 |
| **Use-case Name** | Search for trips | | | |
| **Author** | Nguyen Son Truong | | | |
| **Date** | 2013/01/29 | **Priority** | High | |
| **Actor:**  Customer, Unauthorized user  **Summary:**  User can view list of available trips  **Goal:**  Helps user view all the available trips and choose the most suitable one for them to book.  **Triggers:**  User wants to search for trips.  (User provides search conditions and click button “Search”)  **Preconditions:**  The home page is fully loaded. Lists of all departure/arrival cities and bus types are loaded from database.  **PostConditions:**  Success:   * The search result page is fully loaded. * User can view all of the available trips matching their search conditions.   Failure: System displays error message. Content of error message will be specified in Exception section.  **Main Success Scenario:**   |  |  | | --- | --- | | Actor Actions | System Response | | 1. User chooses type of trips, departure/arrival cities, number of passengers, departure date, arrival date and clicks on button “Đặt vé” [Alternative 1][Alternative 2][Exception 1] [Exception 2]  3. Click “Tiếp tục” button. | 2. Display search result page with details of trips matching search condition. [Exception 3]   * Departure city [Điểm khởi hành: string, label] * Arrival city [Điểm kết thúc: string, label] * Departure time [Giờ đi: DateTime, label] * Arrival time [Giờ đến: DateTime, label] * Fare [Giá vé: double, label] * Select [Chọn chuyến: radio button] * Continue [Tiếp tục: button]     4. Display seat selection page. |   **Alternative Scenario:**   |  |  | | --- | --- | | Actor Actions | System Response | | [Alternative 1]  1. User chooses on “Vé khứ hồi” radio button.  [Alternative 2]  1. User chooses “Vé một chiều” radio button | 2. Display “Ngày về” textbox.  2. Hide “Ngày về” textbox. |   **Exceptions:**   |  |  | | --- | --- | |  |  | | Actor Actions | System Response | | [Exception 1]  1. Departure and arrival cities are the same  [Exception 2]  1. Arrival date is earlier than departure date  [Exception 3]  1. There are no available trips satisfied the conditions | 2. Show info message: “Trạm khởi hành và trạm kết thúc không phù hợp, xin vui lòng chọn lại.”  2. Show info message: “Ngày đi phải nhỏ hơn ngày về, xin vui lòng chọn lại.”  2. Show info message: “Xin lỗi quý khách, hiện tại không có chuyến đi nào phù hợp với yêu cầu của quý khách.” |   **Relationships:**  <None>  **Business Rules:**  User can only reserve for a trip for maximum of 30 days before departure date.  Maximum number of passengers for each reservation is 5. | | | | |

3.2.3. Software System Attributes

3.2.3.1 Reliability

3.2.3.2 Availability

3.2.3.3 Security

3.2.3.4 Maintainability

3.2.3.5 Portability

3.2.3.6 Performance

**3.3. Entity Relationship Diagram or Data Structures**

*<Provide the ERD Diagram for the system here. If your team uses a file or in-memory storage facility instead of a database, replace this section by ‘Data Structures’. Note, use only ERD or Data Structures>*

**3.4. Other material (if any)**

**Report No.4: Software Design Description (SDD)**

**4.1. Design Overview**

**4.2. System Architectural Design**

4.2.1 Choice of System Architecture

4.2.2 Discussion of Alternative Designs

4.2.3 Description of System Interface

**4.3. Component Diagram**

**4.4. Detailed Description of Components**

**4.4.1. CRC Cards (Class-Responsibility-Collaborators)**

4.4.1.*n.* Component-*n*

4.4.1.n.1. Class Diagram

4.4.2.n.2. Class Diagram Explanation

*<Provide a brief explanation of the class diagram above. You do not need to explain “obvious” parts of your class diagram>*

4.4.3.n.3. Algorithms of important methods in each class, specified in pseudo code or by Flow-Chart

**4.5. Sequence Diagram**

**4.6. User Interface Design**

4.6.1 Description of the User Interface

4.12.1.1 Screen Images

4.12.1.2 Objects and Actions

**4.7. Database Design or Data Structures**

*<Provide the detailed database design for the system here. If your team uses a file or in-memory storage facility instead of database, remove this section; use the ‘Data Structures’ section. >*

**4.8. Other material (if any)**

**Report No.5: Software Test Documentation (STD)**

**5.1. Introduction**

1.1 System Overview

1.2 Test Approach

**5.2. Test Plan**

2.1 Features to be tested

2.2 Features not to be tested

2.3 Testing Tools and Environment

**5.3.** **Test Cases**

3.*n* Case-*n*

3.*n*.1 Purpose

3.*n*.2 Inputs

3.*n*.3 Expected Outputs & Pass/Fail criteria

3.*n*.4 Test Procedure

**5.5. Checklists**

5.5.1. Checklist of Validation

*<Put the checklist here. Describe how it is used and the resulted checklist>*

5.5.2. Submission Checklist

*<Put the checklist here. Describe how it is used and the resulted checklist>*

**5.5. Other material (if any)** (including appendix A)

APPENDIX A. TEST LOGS

A.n Log for test *n*

A.n.1 Test Results

A.n.2 Incident Report

**Report No.6: Software User’s Manual**

**6.1. Installation Guide**

**6.2. User’s Guide**

**6.3. Other [Optional]**

**Appendix**