|  |
| --- |
| eProject Document |
|  |
| **Work to Worker System**   |  |  | | --- | --- | | **Group 1** | | | **Group Member** | Nguyễn Văn Cường <C00541>  Bùi Hồng Hải<C00526>  Nguyễn Nhất Linh<C00538>  Lưu Thế Thông<C00542>  Trần Trung Hiếu<C00549> | | **Instructor** | Lê Thị Hảo | | **Batch** | C1101G | | **Semester** | 4 | |
|  |

- Hanoi, 04/2013

# Acknowledgements

First of all, we would like to express our appreciation to all people who helped us to complete the eProject with a lot of idea and helpful advices, especially Mrs.Hao (haolt@fpt.aptech.ac.vn) who explained and gave us detailed instructions. Last but not least, we would like to thank all my classmates for their valuable suggestions and constructive feedback.

# Synopsis

The application is to be developed for Java platform using JDK 7.0, Netbean 7.2 and SQL Server 2008 as back-end.

The application should perform the following functions:

* Store information for managing in the database.
* Register customer and worker, customer and worker can login and manage account (CV of employees and job advertisement of employers).
* Searching system (include jobs for worker and labor for customer)
* Feedback and mail system (job seekers/ workers/ employees and employers can contact each other to exchange information)
* Administrator (manage all account and can backup/restore data)

# Problem Definition

## Introduction

Objective of this system is to providing a work to worker. Solution focused on the workers of unorganized sector includes mason, carpenters, gardener, painter etc. Work to Worker is a Web-based solution through which worker (s) registered themselves for a specific skill. Using this system general public or organized sector user can select the worker(s) as per their need. At the time of worker selection he/she can view the skill, references given by those who have taken their service in the past, area (worker location) and availability of a particular worker(s). System sends SMS to a selected worker(s) regarding work and customer details. Worker(s) confirms either through phone or this system and either fixed up meeting or work start date. Organized sector user or general public can rate worker skill, charges, particularity about time, dedication, behavior, habits etc through this system.   
Users can put their demands regarding particular skill workers along with project location, and project details.

## Goals of the system

Work to Worker is a Web-based solution will have been developed to achieve these goals:

* Providing a work to worker.
* Focused on the workers of unorganized sector.
* General public or organized sector user can select the worker(s) as per their need.
* Worker selection he/she can view the skill, references given by those who have taken their service in the past, area (worker location) and availability of a particular worker.
* System sends SMS to a selected worker(s) regarding work and customer details.
* Worker(s) confirms either through phone or this system and either fixed up meeting or work start date.
* Organized sector user or general public can rate worker skill, charges, particularity about time, dedication, behavior, habits etc through this system.
* Users can put their demands regarding particular skill workers along with project location, and project details.

## Functional goals

* Secure Registration of users (workers) and customers (general public/ organize sector user) with any type of centralized security mechanisms (election card, passport, pan card etc.).
* Workers are classified into various groups & categorized namely aware skilled, semiskilled, etc...
* State can be modified by feedback system & workers rating evaluation algorithms.
* Profile management facilities for both laborers and customers.
* Chat/mail/SMS services (preferably multilingual)
* Customer can make a community for reference purpose etc.
* Customer can give their feedback for worker(s) who worked for them.
* Worker can view details information project (organized sector) in particular location.
* Worker can maintain the work calendar which should be exported to txt; xls or pdf format, there must be mechanisms for face-to-face meeting with customer.
* Local language support.
* Administrator able to generate various reports graphs on strategic data which should be exported on xsl or csv format, also able to backup/recovery data.
* Facilitate appropriate communication between all stakeholders - Discussion forum/chat/mail/polls
* User Manual

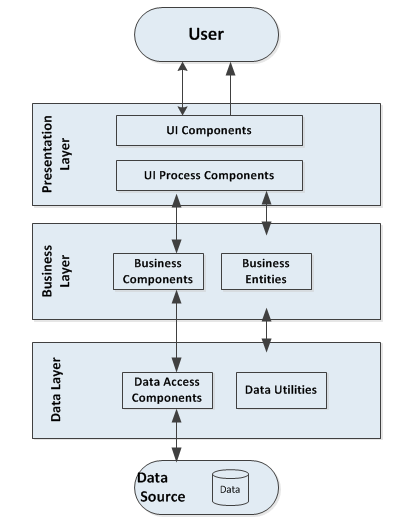
## Development Environment

* + Hardware and OS Requirements:
    - A minimum computer system that will help you access all the tools in the courses is a Pentium 166 or better.
    - 256 Megabytes of RAM or better.
    - HDD: 4GB or later.
    - Windows 2000 or later.
* Software Requirements
  + Java Virtual Machine/ J2EE server
  + Notepad/Java editor
  + j2sdk1.4.1\_02 (or later).
  + EJB Dev Kit
  + Java enabled web server
  + JSP / Servlets De
  + Subversion SVN
  + SQL Server 2008

# Requirement Analysis

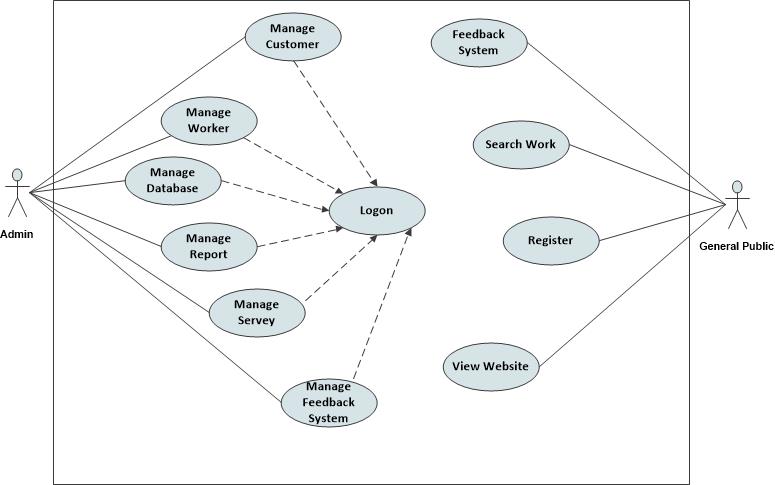
## Architecture and Design of the Project

The application will have three-tier architecture:

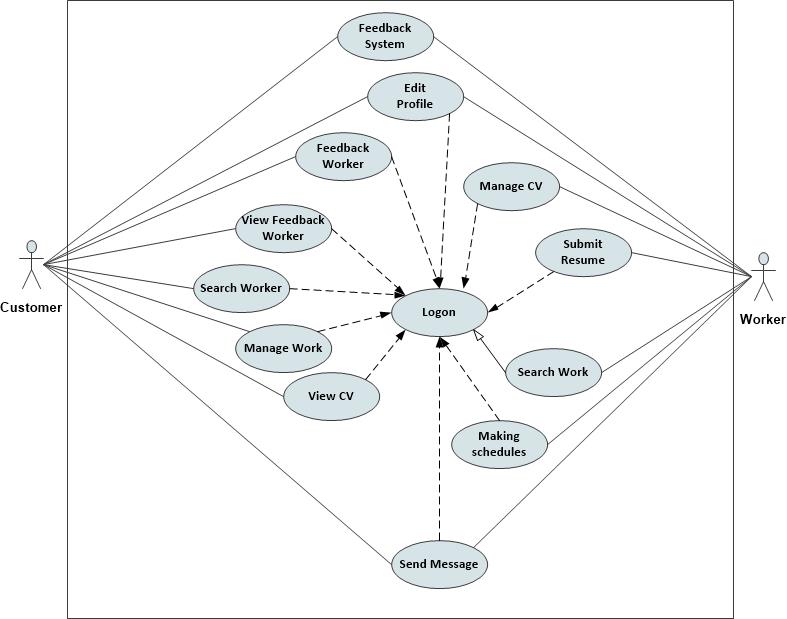


***Figure 1: General System Architecture***

## Use case Diagram



***Figure 2.1: Work to Worker Use Case: Admin & General Pubic***

******

***Figure 2.2: Work to Worker Use Case: Customer & Worker***

### 2.1 List of system's actor business

|  |  |  |
| --- | --- | --- |
| **No** | **Actor** | **Description** |
| 1 | Administrator | A person who can manage Customer, Worker, Database, Report, Survey, Feedback System. |
| 2 | Worker | A person who can use Feedback System, Edit profile, Manage CV, Submit Resume, Search Work, Make Schedule, Send Message. |
| 3 | Customer | A Person who can use Feedback System, Edit profile, Feedback Worker, Search Worker, View Feedback Worker, Manage Work, View CV, Send Message. |
| 4 | General Public | A Person who use Feedback System, Search Work, Registration, View Website. |

### List of use case bussniness

|  |  |  |
| --- | --- | --- |
| **No** | **Use case** | **Description** |
| 1 | Register | Customer enter their information to become member |
| 2 | Logon | Customer, Admin logon to perform the functions allowed. |
| 3 | View product | Customer can see information about products |
| 4 | View seminals,lectures | Customer can see the seminars and lectures |
| 5 | Search | Customer can search information about products, seminars and lectures |
| 6 | Feedback | Customer can send a feedback about products, seminars and lectures |
| 7 | Manage shopping cart | Customer can manage their shopping cart (add or remove items) |
| 8 | Edit profile | Customer or admin can change their profile’s information |
| 9 | View information of website | Customer can see information of the website (eg:address, number contact, email etc.) through contact us or about us |
| 10 | Manage seminals,lectures | Admin can manage the seminars and lectures (edit, delete and update) |
| 11 | Manage report | Admin manage business reports. |
| 12 | Manage feedback | Admin can get feedback from customer and answer their questions. |
| 13 | Manage product | Admin manage products (remove or add new items to website) |
| 14 | Manage members | Admin manage the list of members |
| 15 | Manage information of website | Admin manage website’s details |
| 16 | Manage product introduction | Admin manage product’s details |

### Use Case Specification

|  |
| --- |
| **Use Case Specification** |
| **1.1 Account Details:** Register  **1.2 Use case number:** UC1  **1.3 Actor:** User  **1.4 Brief description:** This use case describes how User can register new Account.  **1.5 Flow Events:**  **1.5.1. Basic flow:**  -This use case starts when User wishes register new Account on website.  -The system requests that the actor enter register information. This include:  User Name  Password  Full Name  Gender  Birth of date  Address  Email  -The User enters register information.  -The system validate User’s register information.  -The system update User’s register information to database.  -The system display register success message and form login.  **1.5.2 Alternative Flows:**  **1.5.2.1. Invalid register information:**  If in basic flow, User entered invalid register information, system display error message . Use can choose retype register information which invalid or cancel register, at which point the use case ends.  **1.6. Special requirements**  None.  **1.7. Pre-Conditions**  None.  **1.8. Post-Conditions**  If use case was successful, User’s register information is insert to database. If not, the system state is unchanged.  **1.9. Extension Points**  None. |

|  |
| --- |
| **Use Case Specification** |
| **2.1 Accout details:** Logon  **2.2 Use case number:** UC2  **2.3.Actor:** User, Admin  **2.4.Brief description:** This use case describes how an Actor logon to the website.  **2.5. Flow Events:**  **2.5.1 Basic flow:**  - This use case starts when the actor want to Logon to the website.  -The system requests that the actor enter his/her user name and password.  -The Actor enters his/her user name and password.  -The system validates the entered user name and password and logs the actor into the system.  **2. 5. 2. Alternative Flows**  **2. 5. 2. 1. Invalid Username/Password**  If in the Basic Flow, the actor enters an invalid user name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.  **2. 6. Special Requirements**  None.  **2. 7. Pre-Conditions**  None.  **2. 8. Post-Conditions**  If the use case was successful, the actor is now logged into the website. If not, the system state is unchanged.  **2. 9. Extension Points**  None |

|  |
| --- |
| **Use Case Specification** |
| **3.1.Account details:** View information of website  **3.2.Use case number:** UC3  **3.3.Actor:** User  **3.4.Brief description:**  This use case describes how a user view information of website.  **3.5.Flow events:**  **3.5.1.Basic flow:**  - This use case starts when the actor wishes to View information of website.  - The system displays information of website. This include:  **+Contact us:** include details of the contact.  **+About us:** include details of the clinic.  - The user can choose to continue view information of website or cancel to view, at which point the use case ends.  **3. 6. Special Requirements**  None.  **3. 7. Pre-Conditions**  None.  **3. 8. Post-Conditions**  If the use case was successful, the user can view information of website . If not, the system state is unchanged.  **3. 9. Extension Points**  None |

|  |
| --- |
| **Use Case Specification** |
| **4.1.Accout details:** Edit profile  **4.2.Use case number:** UC4  **4.3.Actor:** User, Admin  **4.4.Brief description:**  -This use case allows the User/Admin to manage Account profile and change password.  -This include Edit profile, Change password.  **4.5.Flow events:**  **4.5.1.Basic flow:**  -This use case starts when the Actor wishes to edit his/her profile, change password.  -The system request that Admin/User specify the function he/she would like perform(Either edit profile, change password)  -Once Admin/User provides the requested information, one the sub-flows is executed.  -If User selected “edit profile”; the **Edit profile** sub-flow is executed.  -If Admin/User selected “change password”; the **Change password** sub-flow is executed.  **4.5.1.1.Edit profile:**  -The system shows User profile from database and requests that Actor can choose which information he/she want edit. This include:  Name  Gender  Birth date  Address  Phone number  Email  -The User change which information he/she want edit and submit new information.  -The system validate new information of User.  -The system update new information of user to database.  **4.5.1.2.Change password:**  -The system requests that the Actor enter his/her password, new password, retype new password.  -The actor enters his/her password, new password, retype new password.  -The system validate Actors password, new password, retype new password.  -The system update new Actors password to database.  **4.5.2.Alternative flow:**  **4.5.2.1.Invalid new information:**  If in **“Edit profile”** sub-flow , the User enters an invalid information, the system displays an error message. The actor can choose to either return to the beginning of the **“Edit profile”** sub-flow or cancel Edit profile, at which point the use case ends.  **4.5.2.1.Invalid password:**  If in **“Change password”** sub-flow , the User enters an invalid password, new password, retype new password, the system displays an error message. The actor can choose to either return to the beginning of the **“Change password”** sub-flow or cancel Change password, at which point the use case ends.  **4.6.Special requirements:**  **-**Admin only has permission to use **“Change password”** of his/hers.  **4.7.Pre-Conditions:**  -The actor must be logged into website before this use case begins.  **4.8.Post-Conditions:**  -If use case was successful, new information, new password is updated to database. If not, the system state is unchanged.  **4.9.Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **5.1.Accout details:** View product  **5.2.Use case number:**  UC5  **5.3.Actor:** User  **5.4.Brief description:**  This use case describes how Actor can view product information.  **5.5.Flow events:**  **5.5.1.Basic flow:**  - This use case starts when the actor wishes to View product information.  - The system displays product information. This include:  **+Scientific:**  **image**  name  product\_code  manufacture  price  description  “add product to shopping cart” button.    +Medical:  Image  Name  Product\_code  Price  Description  “add product to shopping cart” button.  **5.5.2.Alternative flow:**  - If in basic flow, User choose “add product to shopping cart” button, it starts sub-flow **“Add item”** in use case **“Manage shopping cart”.**  - The user can choose to continue view product information or cancel to view, at which point the use case ends.  **5.6.Special requirements:**  Not include “add product to shopping cart” button if product not for sales.  **5.7.Pre-Conditions:**  None.  **5.8.Post-Conditions:**  If the use case was successful, the user can view product information. If not, the system state is unchanged.  **5.9.Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **6.1.Accout details:** View seminars, lectures  **6.2.Use case number:** UC6  **6.3.Actor:** User  **6.4.Brief description:**  This use case describes how User can view seminars, lectures.  **6.5.Flow events:**  **6.5.1.Basic flow:**  - This use case starts when the User wishes to View seminars, lectures information.  - The system displays seminars, lectures information. This include:  +Seminars, lectures name  +description  +link to join seminars, lectures  - The user can choose to continue seminars, lectures information or cancel to view, at which point the use case ends.  **6.6.Special requirements:**  None.  **6.7.Pre-Conditions:**  None.  **6.8.Post-Conditions:**  If the use case was successful, the user can view seminars, lectures. If not, the system state is unchanged.  **6.9.Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **7.1.Accout details:** Search product  **7.2.Use case number:** UC7  **7.3.Actor:** User  **7.4.Brief description:**  This use case describes how User can Search product.  **7.5.Flow events:**  **7.5.1.Basic flow:**  -This use case starts when the User wishes to Search product.  -The system requests that the User enter keyword to search.  -The User enters keyword and submit.  -The system uses User’s keyword to search data in database.  -The system displays result match with User’s keyword.  **7.5.2.Alternative flow:**  **7.5.2.1.Result not found:**  -If in basic flow, system can’t found result match witch User’s keyword, the system display can’t found product. User can choose to either return to beginning of the “**Search Product”** basic-flow or cancel Search product, at which point the use case ends.  **7.6.Special requirements:**  None.  **7.7.Pre-Conditions:**  None.  **7.8.Post-Conditions:**  If the use case was successful, the User can view result of **“Search product”** basic-flow. If not, the system state is unchanged.  **7.9.Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **8.1.Accout details:** Feedback  **8.2.Use case number:** UC8  **8.3.Actor:** User  **8.4.Brief description:**  -This use case describes how User can Feedback to the system.  **8.5.Flow events:**  **8.5.1.Basic flow:**  -This use case starts when the User wishes to Feedback information to the system.  -The system requests that User enter the Feedback information and submit.  -The system insert Feedback information to database, at which point the use case ends.  **8.6.Special requirements:**  None.  **8.7.Pre-Conditions:**  -The actor must be logged into website before this use case begins.  **8.8.Post-Conditions:**  If the use case was successful, the Feedback information is inserted to database . If not, the system state is unchanged.  **8.9.Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **9.1.Accout details:** Manage shopping cart  **9.2.Use case number:** UC9  **9.3.Actor:** User  **9.4.Brief description:**  **-**This use case allows the User to manage shopping cart information.  -This include add item, edit item, remove item, purchase item, manage order.  **9.5.Flow events:**  **9.5.1.Basic flow:**  -This use case starts when User wishes to Manage his/her shopping cart information.  -The system request that User specify the function he/she would like perform(Either add item, edit item, remove item, purchase item, manage order)  -Once User provides the requested information, one the sub-flows is executed.  -If User selected “edit item”; the **Edit item** sub-flow is executed.  -If User selected “remove item”; the **Remove item** sub-flow is executed.  -If User selected “manage order”; the **Manage order** sub-flow is executed.  -If User selected “checkout”; the **Checkout** sub-flow is executed.  **9.5.1.1.Add item:**  -This sub-flow starts when the User uses use case **“View product”** and choose “add product to shopping cart”.  -The system get information of product in database and requests that User enter quantity of item.  -The User enter quantity of item.  -The system insert information of product and quantity of item to database.  **9.5.1.2.Edit item:**  -The system displays information of product and quantity of item from database.  -The system requests that User enter new quantity of item.  -The User enter new quantity of item.  -The system update new quantity of item to database.  **9.5.Remove item:**  -The system displays information of product and quantity of item from database.  -The system request that User choose which item to remove.  -The User choose item to remove.  -The system delete that item from database.  **9.5.1.4.Manage order:**  -The system displays information of product, quantity of item from database and calculate subtotal.  -The system request that User enter Order information. This include:  Full Name  Email  Phone number  Delivery address  Identification ID  Payment methods  -The User enter Order information and submit.  -The system insert Order information to database.  **9.5.1.5.Checkout:**  - The system displays Checkout Order. This include:  information of product  quantity of item  Order information  subtotal.  -The User can choose cancel or submit Order.  -If User choose submit Order , the system insert Checkout Order to database. If not, the system state is unchanged, at which point the use case ends.  **9.6.Special requirements:** None.  **9.7.Pre-Conditions:** None. |

|  |
| --- |
| **Use Case Specification** |
| **10.1. Account details:** Manage members  **10.2. Use case number:** UC10  **10.3. Actor:** Admin  **10.4. Brief description:**  **-**This use case allows the Admin to manage members.  -This include view list of all members and remove members from website.  **10.5. Flow events:**  **10.5.1. Basic flow:**  -This use case starts when Admin want to manage website’s member.  -The system request that Admin specify the function he/she would like perform (Either view list of members or remove members)  -Once Admin provides the requested information, one the sub-flows is executed.  -If Admin selected “view list of all member”; the **View list members** sub-flow is executed.  -If Admin selected “delete members”; the **Delete member’s account** sub-flow is executed.  **10.5.1.1. View list members:**  - The system shows all member profile from database. This includes:  Name  Gender  Birth date  Address  Phone number  Email  **10.5.1.2. Delete member’s account:**  The system requests that the Admin enter member’s account he want to remove.  **10.5.2. Alternative Flows:**  **10.5.2.1. Invalid member’s account:**  If in the Basic Flow, the Admin enters an invalid member’s account, the system displays an error message. The actor can choose to either return back to see list of all members.    **10.6. Special requirements:**  None.  **10.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **10.8. Post-Conditions:**  If the use case was successful, the Admin can manage members of website. If not, the system state is unchanged.  **10.9. Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **11.1. Account details:** Manage seminars, lectures  **11.2. Use case number:** UC11  **11.3. Actor:** Admin  **11.4. Brief description:**  **-**This use case allows the Admin to manage seminars and lectures.  -This includes add new, edit and remove seminars and lectures.  **11.5. Flow events:**  **11.5.1: Basic flow:**  -This use case starts when Admin want to manage seminars and lectures on website.  -The system request that Admin specify the function he/she would like perform (Either edit, add new and remove seminars and lectures)  -Once Admin provides the requested information, one the sub-flows is executed.  -If Admin selected “edit seminars and lectures”; the **Edit seminars and lectures** sub-flow is executed.  -If Admin selected “add new seminars and lectures”; the **Add new seminars and lectures** sub-flow is executed.  -If Admin selected “remove seminars and lectures”; the **Remove seminars and lectures** sub-flow is executed.  **11.5.1.1: Add new seminars and lectures:**  - This use case starts when the Admin want to add new seminars and lectures information.  - The system displays seminars and lectures form to enter information. This includes:  +Seminars, lectures name  +description  +link to join seminars, lectures  **11.5.1.2: Edit seminars and lectures:**  -This use case starts when the Admin want to edit seminars and lectures information.  **11.5.1.3: Remove seminars and lectures:**  -This use case starts when the Admin want to remove seminars and lectures information.    **11.6. Special requirements:**  None.  **11.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **11.8. Post-Conditions:**  If the use case was successful, the Admin can manage seminars and lectures. If not, the system state is unchanged.  **11.9. Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **12.1. Account details:** Manage report  **12.2. Use case number:** UC12  **12.3. Actor:** Admin  **12.4. Brief description:**  **-**This use case allows the Admin to manage report.  **12.5. Flow events:**  **12.5.1: Basic flow:**  -This use case start when Admin want to manage report on website.  -The system request that Actor enter report information. This include:  *Select date*  *Choose type of report*  -The system statistic data from database base on report information and display to actor.  -The actor can choose either continue view report result, print report result or cancel manage report, at which point the use case ends.  **12.6. Special requirements:**  **12.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **12.8. Post-Conditions:**  If the use case was successful, the Admin can manage report. If not, the system state is unchanged.  **12.9. Extension Points:**  None. |

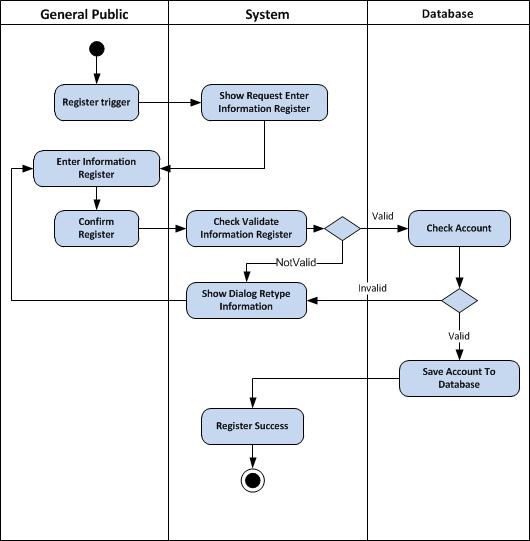
|  |
| --- |
| **Use Case Specification** |
| **13.1. Account details:** Manage website’s information  **13.2. Use case number:** UC13  **13.3. Actor:** Admin  **13.4. Brief description:**  **-**This use case allows the Admin to manage website’s information.  -This includes manage product (edit contact us and edit about us of website’s information).  **13.5. Flow events:**  **13.5.1: Basic flow:**  -This use case starts when Admin want to manage website’s information.  -The system request that Admin specify the function he/she would like perform (Either edit contact and edit about of website’s information)  -Once Admin provides the requested information, one the sub-flows is executed.  -If Admin selected “edit contact us”; the **Edit contact us** sub-flow is executed.  -If Admin selected “edit about us”; the **Edit about us** sub-flow is executed.  **13.5.1.1.Edit contact us:**  -The system displays Contact us information and requests that Actor edit contact us information.  -The actor edit contact us information and submit.  -The system update contact us information to database.  **13.5.1.2.Edit about us:**  -The system displays About us information and requests that Actor edit contact us information.  -The actor edit About us information and submit.  -The system update About us information to database.  **13.6. Special requirements:**  None.  **13.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **13.8. Post-Conditions:**  If the use case was successful, the Admin can manage website’s information. If not, the system state is unchanged.  **13.9. Extension Points:**  None. |

|  |
| --- |
| **Use Case Specification** |
| **14.1. Account details:** Manage product  **14.2. Use case number: UC14**  **14.3. Actor: Admin**  **14.4. Brief description:**  **-**This use case allows the Admin to manage product.  -This includes manage product (add new, edit, delete and search product and add new, update and remove category).  **14.5. Flow events:**  **14.5.1: Basic flow:**  -This use case starts when Admin want to manage product on website.  -The system request that Admin specify the function he/she would like perform (Either add new, edit, delete and search product and add new, update and remove category)  -Once Admin provides the requested information, one the sub-flows is executed.  -If Admin selected “edit product”; the **Edit product** sub-flow is executed.  -If Admin selected “add product”; the **Add new product** sub-flow is executed.  -If Admin selected “remove product”; the **Remove product** sub-flow is executed.  -If Admin selected “add new category”; the **Add new category** sub-flow is executed.  -If Admin selected “update category”; the **Update category** sub-flow is executed.  -If Admin selected “remove category”; the **Remove category** sub-flow is executed.  **14.5.1.1.Edit product:**  -The system request that Actor enter Product\_code.  -The actor enters Product\_code and submit.  -The system search product from database base on Product\_code and display product information to Actor and request that Actor edit product information.  -The Actor edit product information and submit.  -The system update product information to database.  **14.5.1.2.Add new product:**  -The system request that Actor enter product information. This include:  **+Scientific:**  **image**  name  product\_code  manufacture  price  description    +Medical:  Image  Name  Product\_code  Price  Description  -The actor enters product information.  -The system validate product information and insert it to database.  **14.5.1.3.Remove product:**  -The system request that Actor enter Product\_code.  -The actor enters Product\_code and submit.  -The system delete product from database base on Product\_code.  **14.5.1.4.Add new category:**  -The system request that Actor enter category’s name.  -The Actor enters category’s name.  -The system insert new category to database.  **14.5.1.5.Update category:**  -The system request that Actor enter category’s name.  -The Actor enters category’s name.  -The system request that Actor enter new category’s name.  -The Actor enters new category’s name.  -The system update new category’s name to database.  **14.5.1.6.Remove category:**  -The system request that Actor enter category’s name.  -The Actor enters category’s name.  -The system delete category from database base on category’s name.  **14.6. Special requirements:**  None.  **14.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **14.8. Post-Conditions:**  If the use case was successful, the Admin can manage product. If not, the system state is unchanged.  **14.9. Extension Points:**  None. |

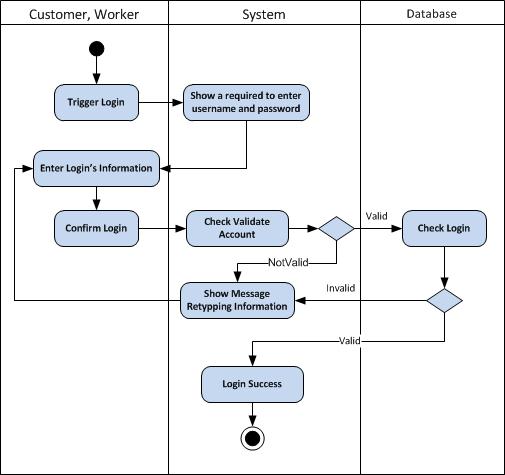
|  |
| --- |
| **Use Case Specification** |
| **15.1. Account details:** Manage feedback  **15.2. Use case number: UC15**  **15.3. Actor: Admin**  **15.4. Brief description:**  **-**This use case allows the Admin to manage feedback.  -This includes manage feedback(view feedback, delete feedback).  **15.5. Flow events:**  **15.5.1: Basic flow:**  -This use case starts when Admin want to manage feedback on website.  -The system request that Admin specify the function he/she would like perform (Either view feedback, report feedback)  -Once Admin provides the requested information, one the sub-flows is executed.  -If Admin selected “view feedback”; the **View feedback** sub-flow is executed.  -If Admin selected “delete feedback”; the **Delete feedback** sub-flow is executed.  **15.5.1.1.View feedback:**  -The system display feedback from database to Actor.  -The Actor can choose either continue view feedback or Delete feedback or cancel view feedback.  **15.5.1.2.Delete feedback:**  -The system request that Actor choose which feedback to delete.  -The Actor choose feedback to delete.  -The system delete feedback from database.  **15.6. Special requirements:**  None.  **15.7. Pre-Conditions:**  The actor must be logged into website before this use case begins  **15.8. Post-Conditions:**  If the use case was successful, the Admin can manage feedback. If not, the system state is unchanged.  **15.9. Extension Points:**  None. |

## Active Diagram

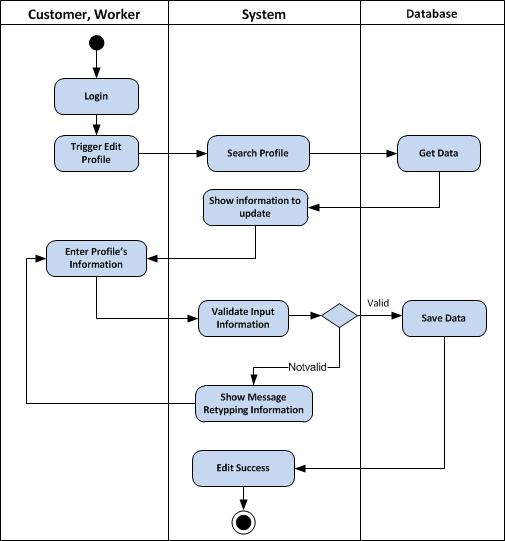
### Register



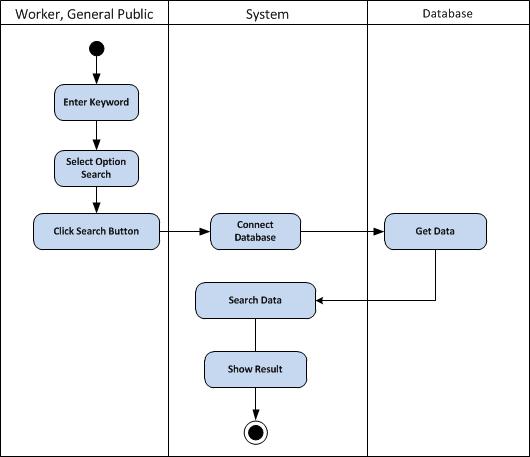
### Login



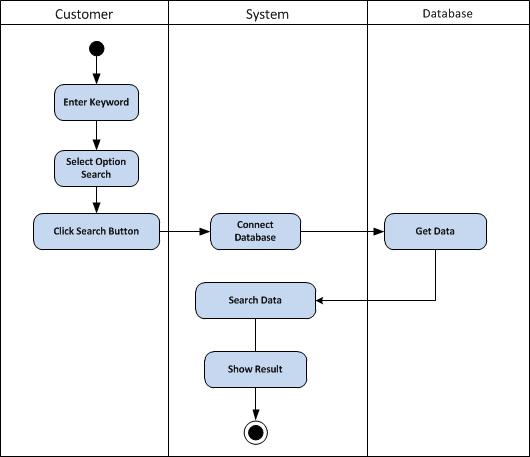
### Edit Profile



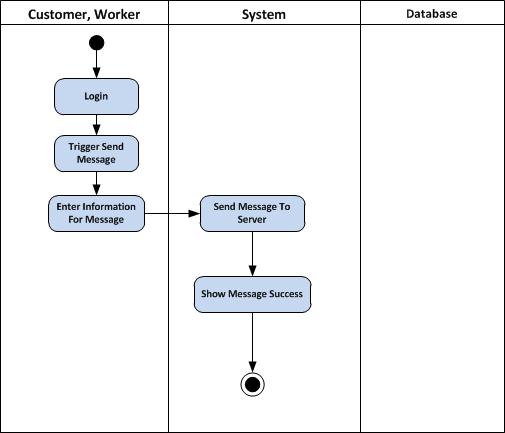
### Search Work



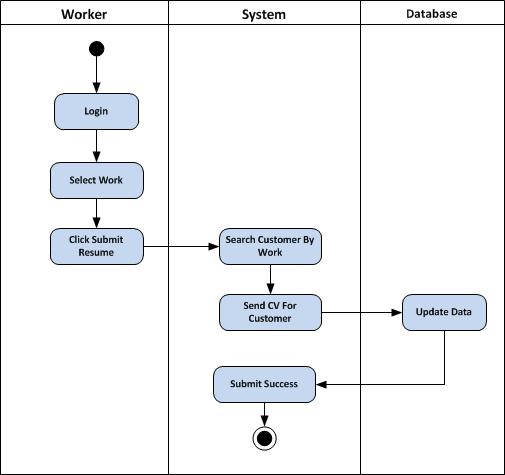
### Search Worker



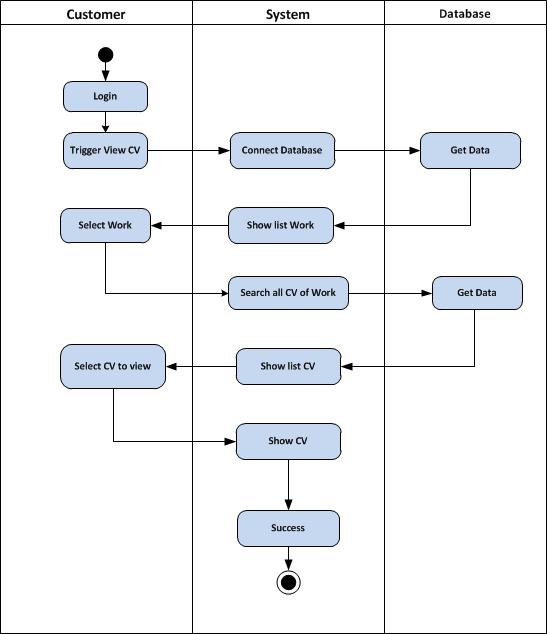
### Send Message



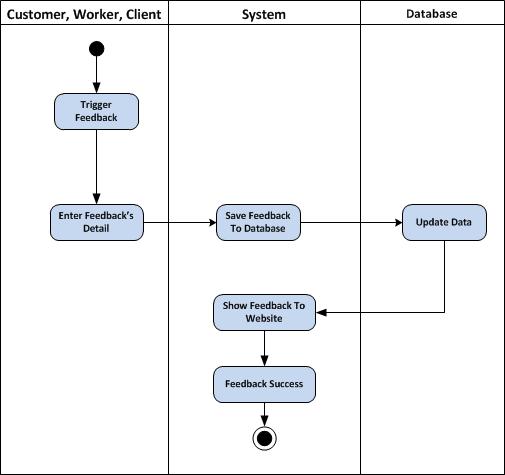
### Submit Resume



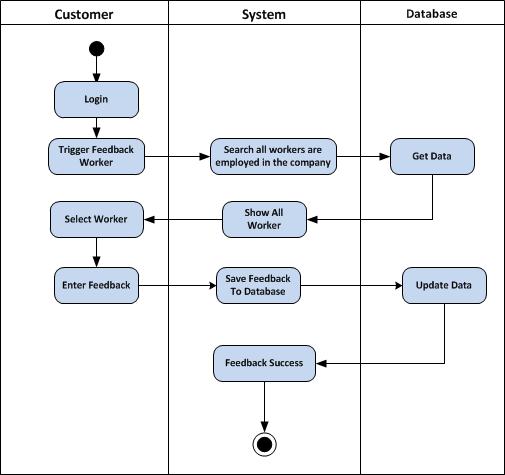
### Accept CV



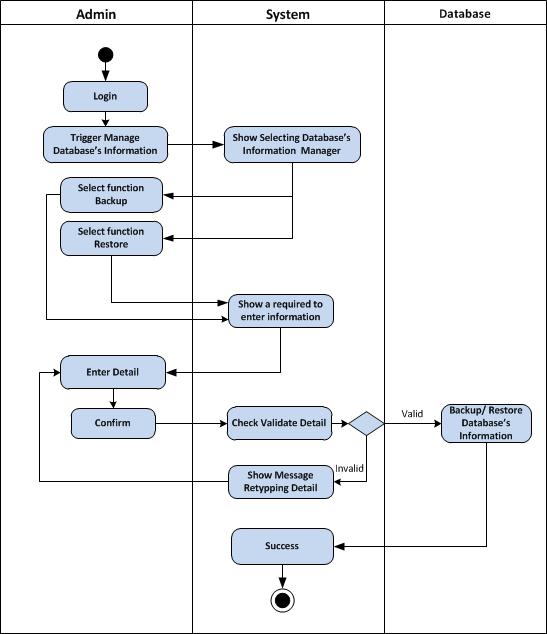
### Feedback System



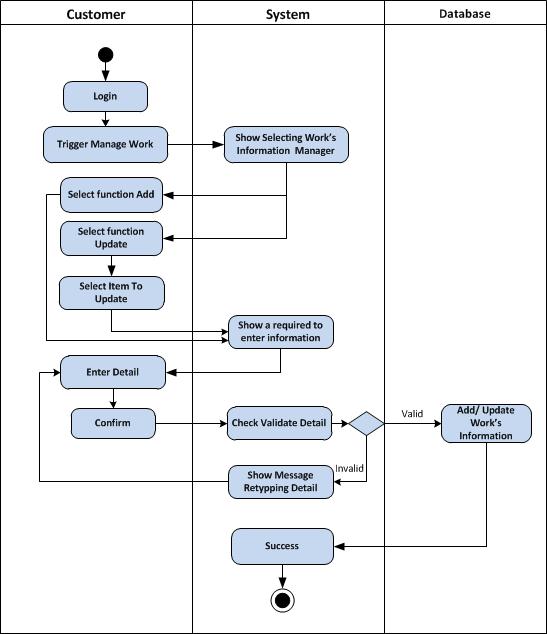
### Feedback Labor



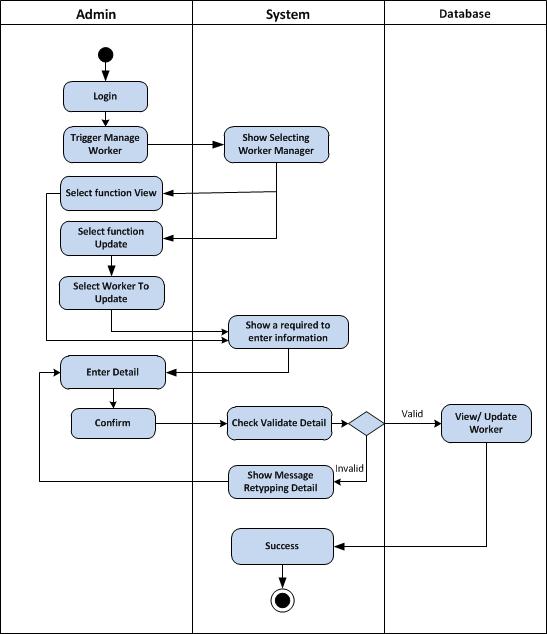
### Manager Database



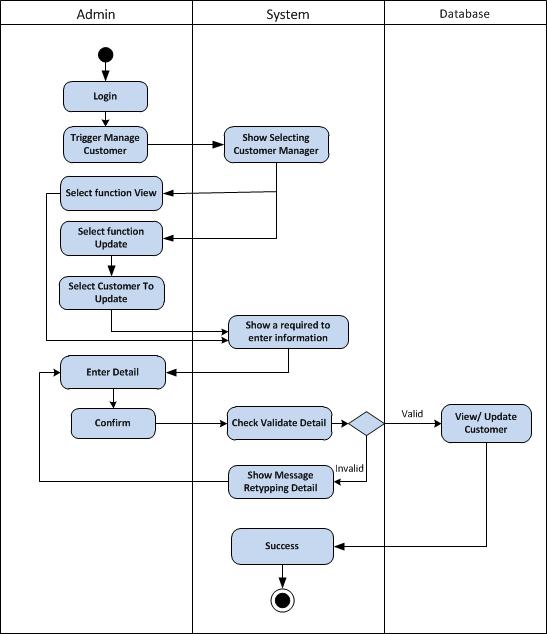
### Manage Jobs



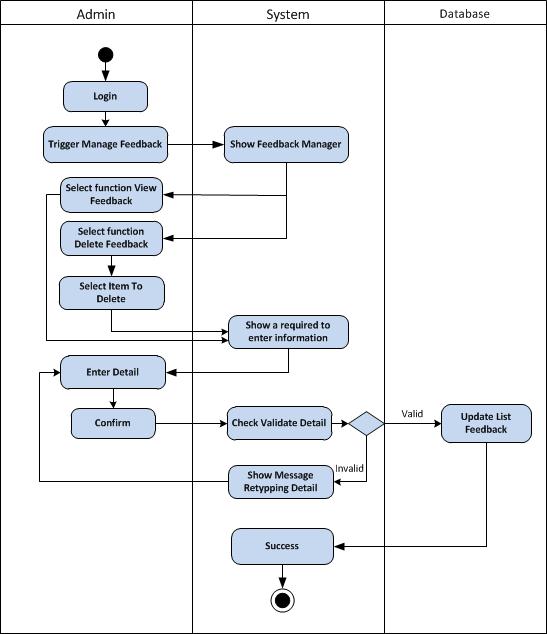
### Manage Employee



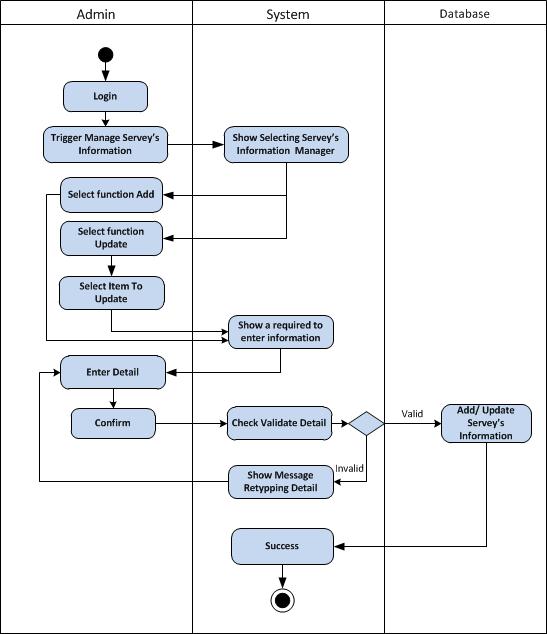
### Manage Customer



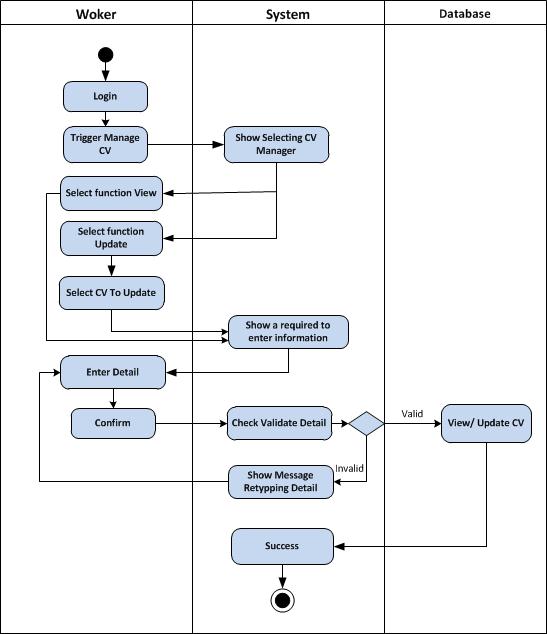
### Manage Feedback



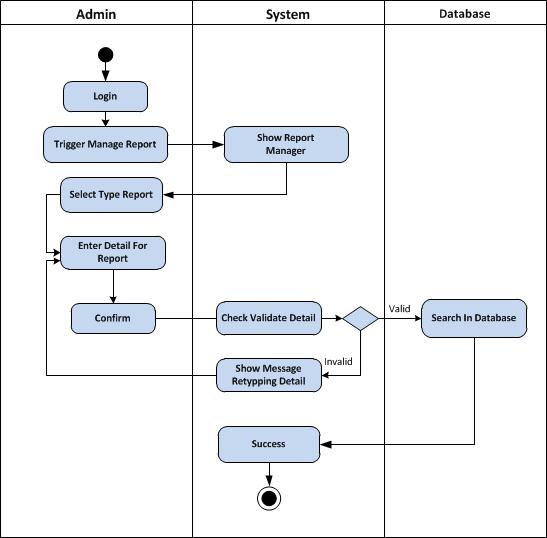
### Manage Survey



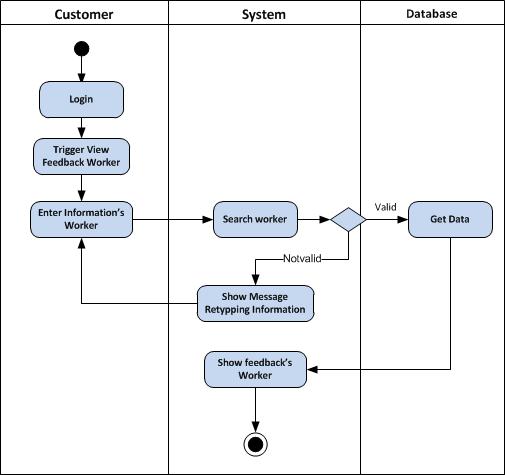
### Manage CV



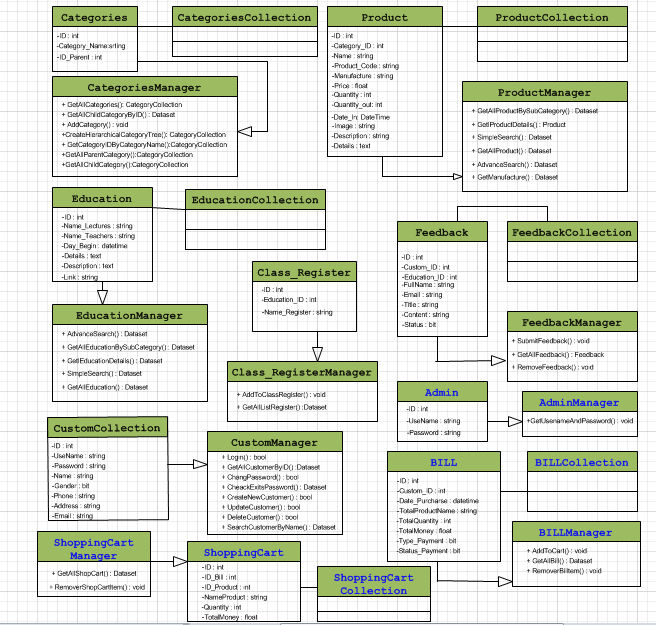
### Manage Report



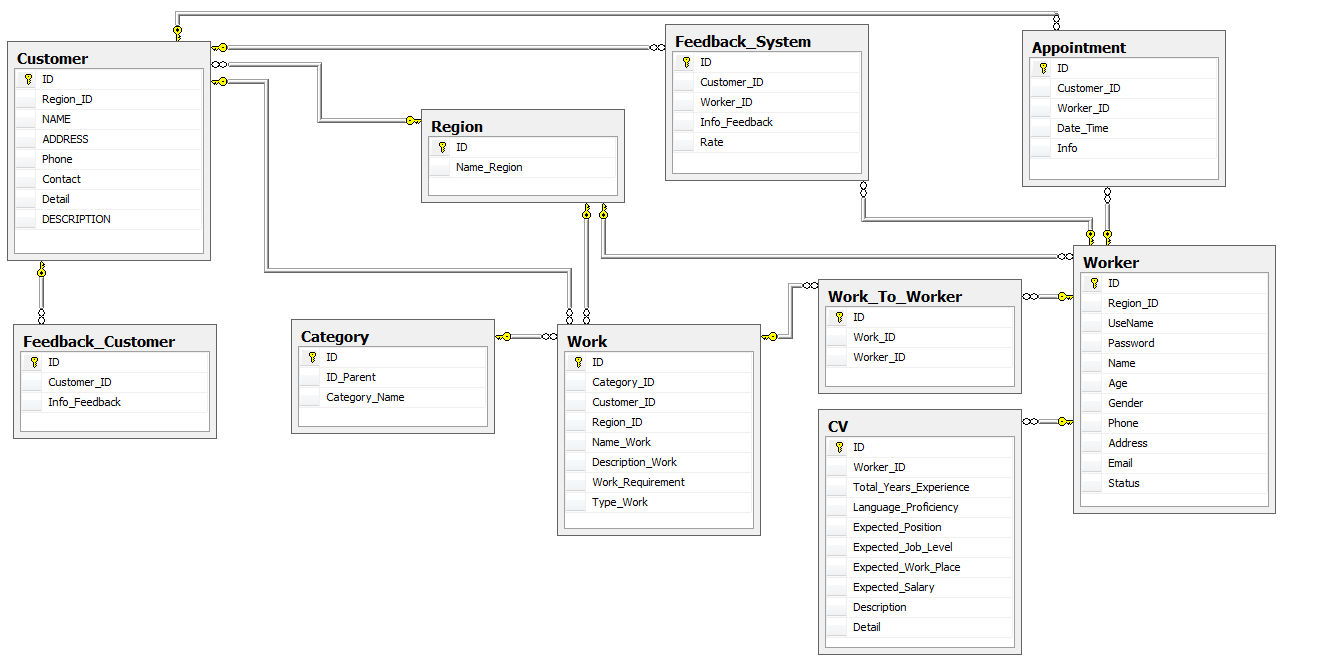
### View Feedback of worker



## Class Diagram



## Entity Relationship(ER) Diagram



### Database Design/ Structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Admin** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the Admin |
| UserName | Nvarchar(50) | No | Unique | Stores the UserName of the Admin |
| Password | Nvarchar(200) | No |  | Stores the password of the account admin |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Feedback** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the Feedback |
| Customer\_ID | int | No | FK | Stores the ID of Customer |
| FullName | Nvarchar(50) |  |  | Stores the fullname’s customer |
| Email | Nvarchar(50) |  |  | Stores the email’s customer |
| Title | Nvarchar(100) |  |  | Stores the title of feedback |
| Content | Nvarchar(200) |  |  | Stores the content of feedback |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Customer** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the Customer |
| UseName | Nvarchar(50) | No |  | Stores the UseName of the Customer |
| Password | Nvarchar(200) | No |  | Stores the password of the Customer |
| Name | Nvarchar(50) |  |  | Stores the Name of the Customer |
| Gender | bit |  |  | Stores the gender of the Customer |
| Phone | Nvarchar(20) |  |  | Stores the phonenumber of Customer |
| Address | Nvarchar(100) |  |  | Stores the address of the Customer |
| Email | Nvarchar(100) |  | Unique | Stores the email of the Customer |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Bill** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the bill |
| Customer\_ID | int |  | FK | Stores the ID of customer |
| Date\_Purcharse | Date Time |  |  | Stores Date of purchase |
| TotalQuantity | int |  |  | Stores the total quantity |
| TotalMoney | float |  |  | Stores the total money |
| Type\_Payment | Nvarchar |  |  | Stores the Type payment |
| Status\_Payment | bit |  |  | Stores the status payment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Product** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int |  | PK | Stores the ID of the Product |
| Category\_ID | ntext |  | FK | Stores the ID of Category |
| Name | Nvarchar(200) |  |  | Stores the Name of Product |
| Product\_Code | Nvarchar(200) |  |  | Stores the code of product |
| Manufacture | Nvarchar(200) |  |  | Stores the name of Manufacture |
| Price | Float |  |  | Stores the price of product |
| Quantity\_In | int | no |  | Stores the Quantity in of Product |
| Quantity\_Out | int |  |  | Stores the Quantity out of Product |
| Date\_In | Date Time |  |  | Stores the Date In of Product |
| Image | Nvarchar(200) |  |  | Stores the image of product |
| Description | Ntext |  |  | Stores the description of product |
| Details | Ntext |  |  | Stores the details of product |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Categories** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the Categories |
| Category\_Name | Nvarchar(200) | No |  | Stores the name of Categories |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Class\_Register** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int |  | PK | Stores the ID of the Class\_Register |
| Education\_ID | int |  | FK | Stores the ID of Education |
| Name\_Register | Nvarchar(200) | No |  | Stores the Register |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Education** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of Education |
| Category\_ID | int | No | FK | Stores the ID of Category |
| Name\_Lectures | Nvarchar(300) |  |  | Stores the Name’s Lectures |
| Name\_Teachers | Nvarchar(300) |  |  | Stores the Name’s Teachers |
| Day\_Begin | Datetime |  |  | Stores the day starting lectures |
| Detail | Ntext |  |  | Stores the detail of lectures |
| Description | Ntext |  |  | Stores the description of lectures |
| Link | Nvarchar(500) |  |  | Stores the link to view details |

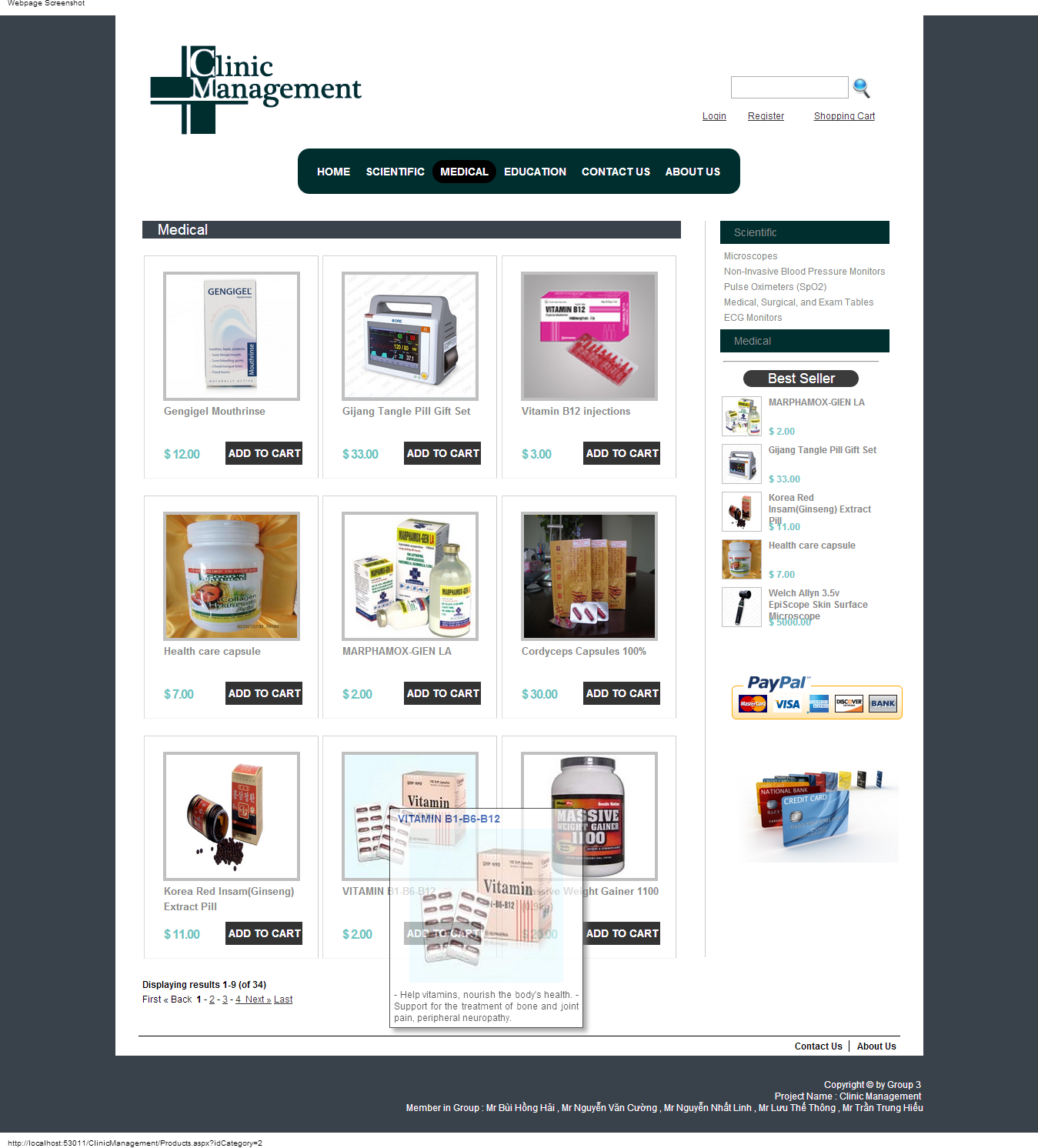
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: ShoppingCart** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | int | No | PK | Stores the ID of the Shopping Cart |
| ID Bill | int | No | FK | Stores the ID Bill |
| ID Product | Int | No | FK | Stores the ID Product |
| Quantity | Int | No |  | Stores the Quantity Product |
| TotalMoney | Float | No |  | Stores the Total Money of product |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table: Credit Card** | | | | |
| ***Field Name*** | ***Data Type*** | ***Null*** | ***Key*** | ***Description*** |
| ID | Int | No | PK | Stores the ID of the Credit Card |
| ID\_Customer | Int | No | FK | Stores the ID of the Customer |
| Card\_Type | Nvarchar(50) | No |  | Stores the Type of the Credit Card |
| Card\_Number | Nvarchar(25) | No |  | Stores the Number of the Credit Card |
| Exp\_Month | Int | No |  | Stores the Month on the Credit Card |
| Exp\_Year | Int | No |  | Stores the Yeah on the Credit Card |
| Modified\_date | Datetime | No |  | Stores the Modified of the Credit Card |

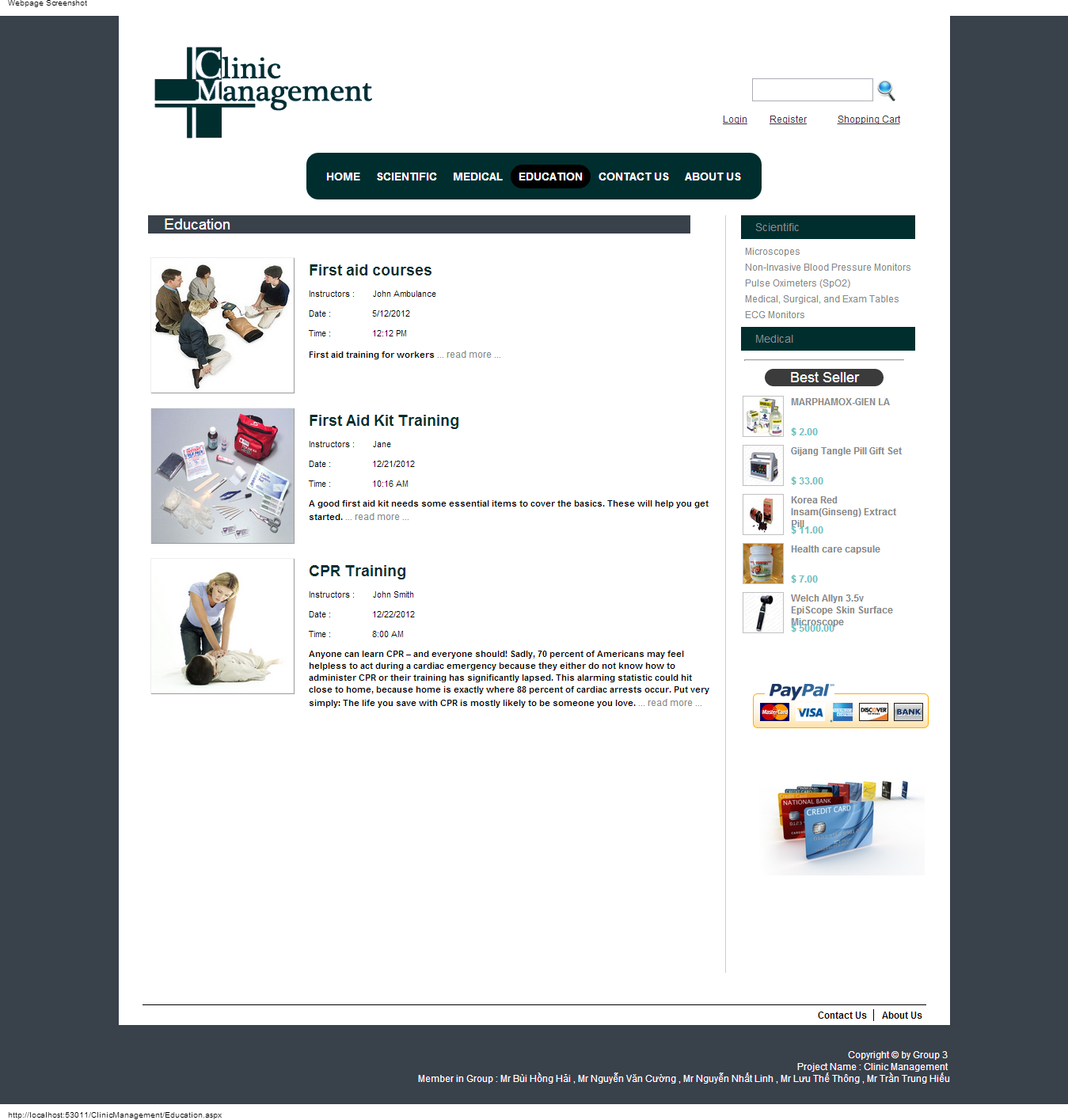
## Screen Shot’s

### Home

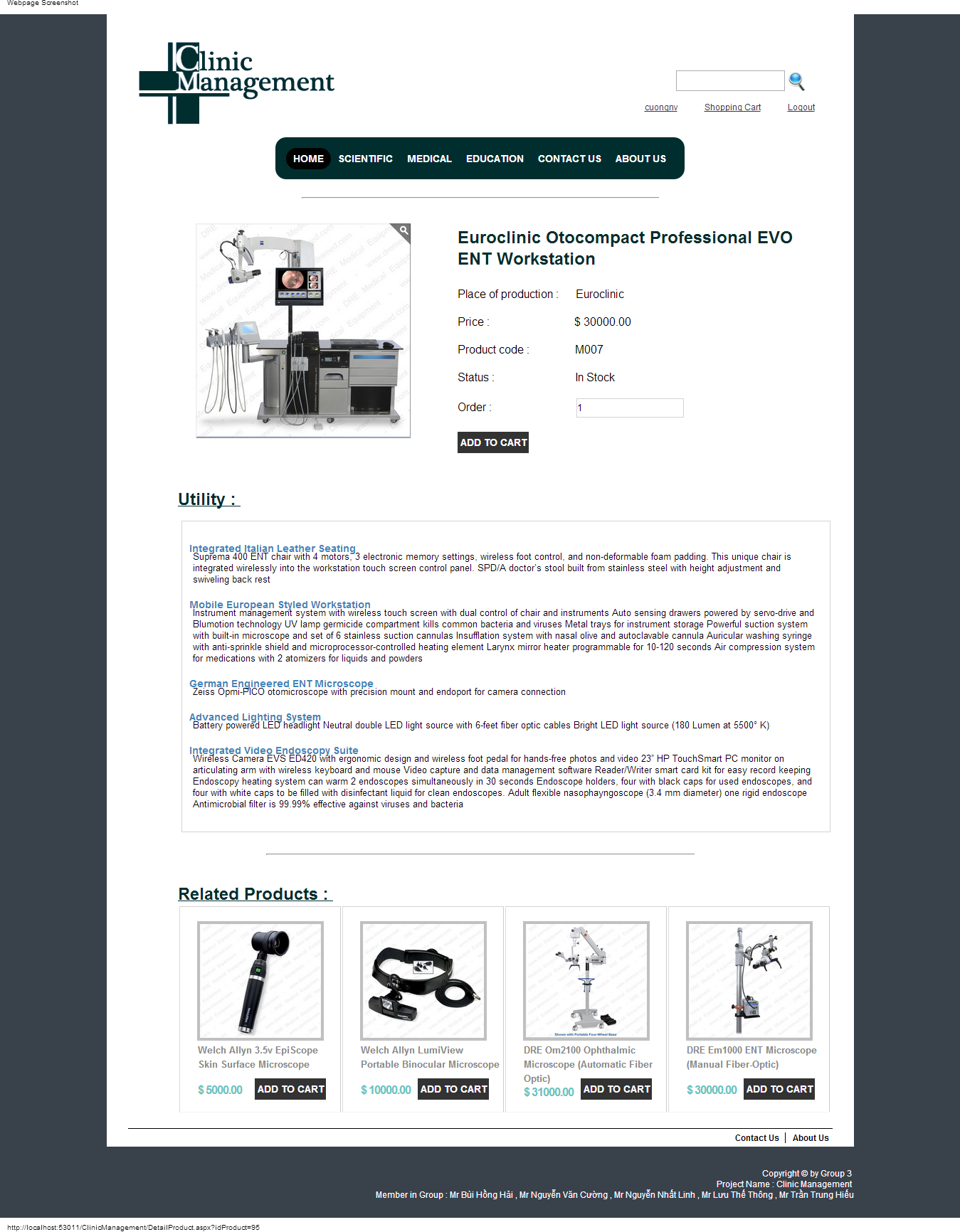
### Product



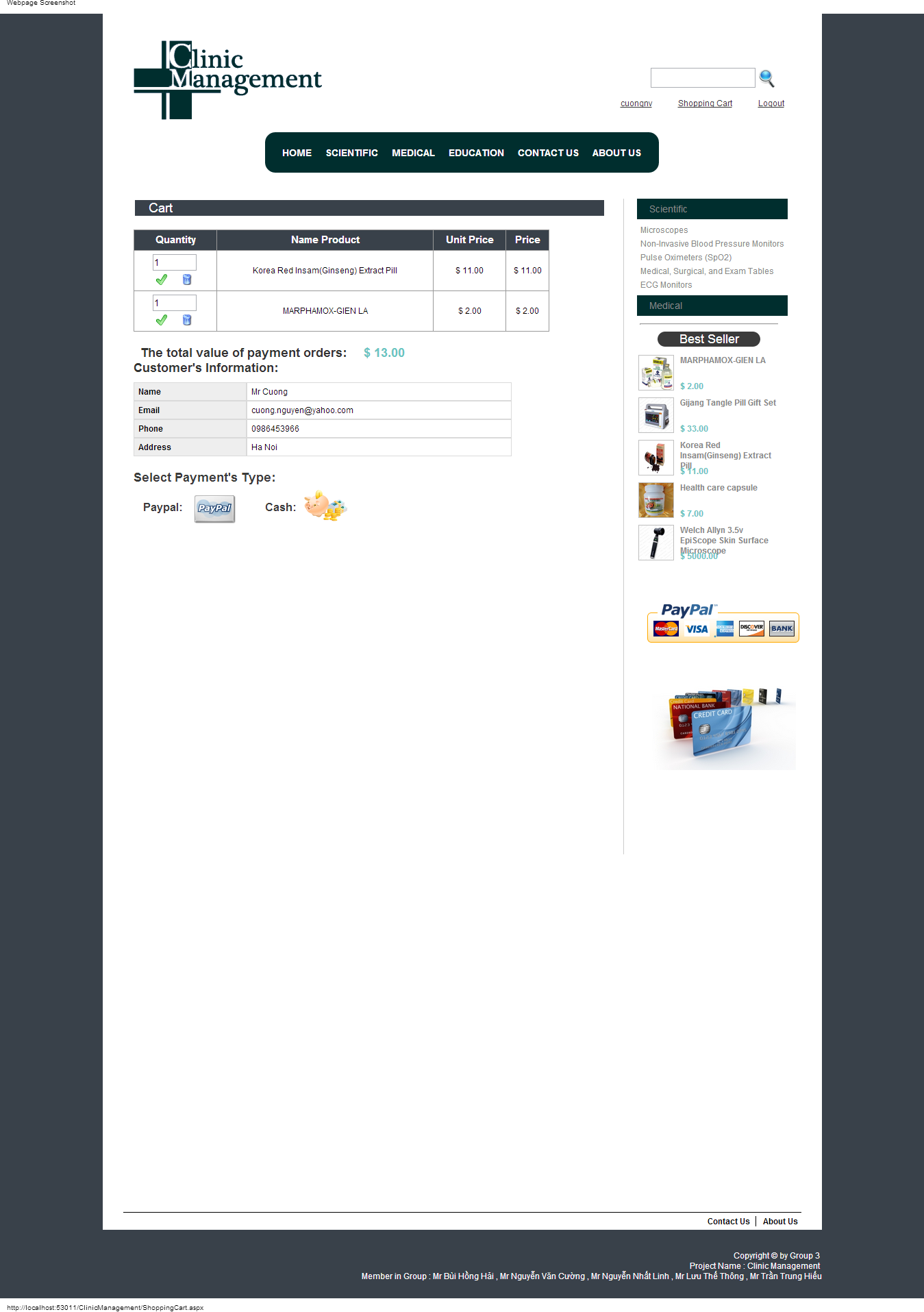
### Education



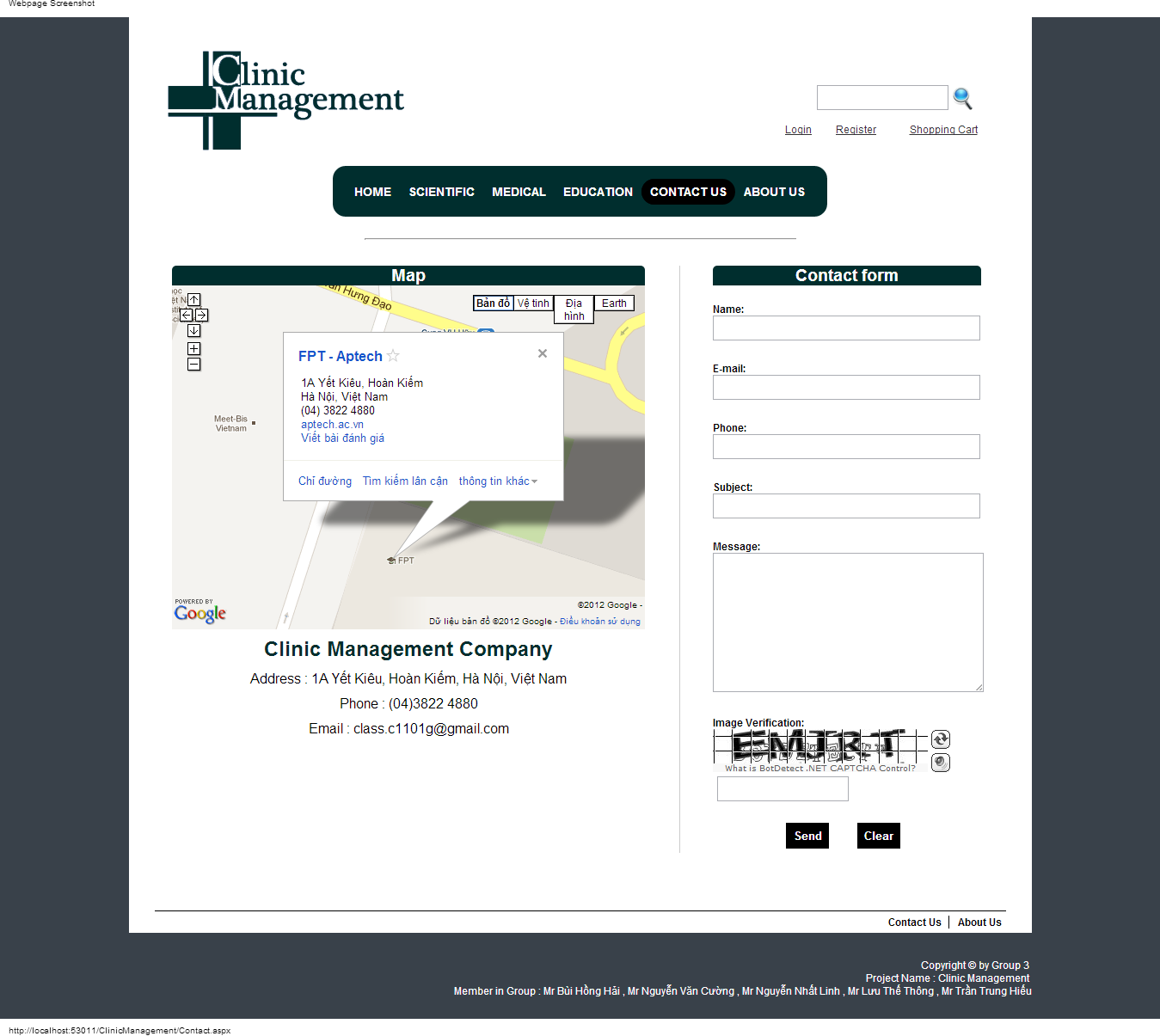
### Product Detail



### Shopping Cart



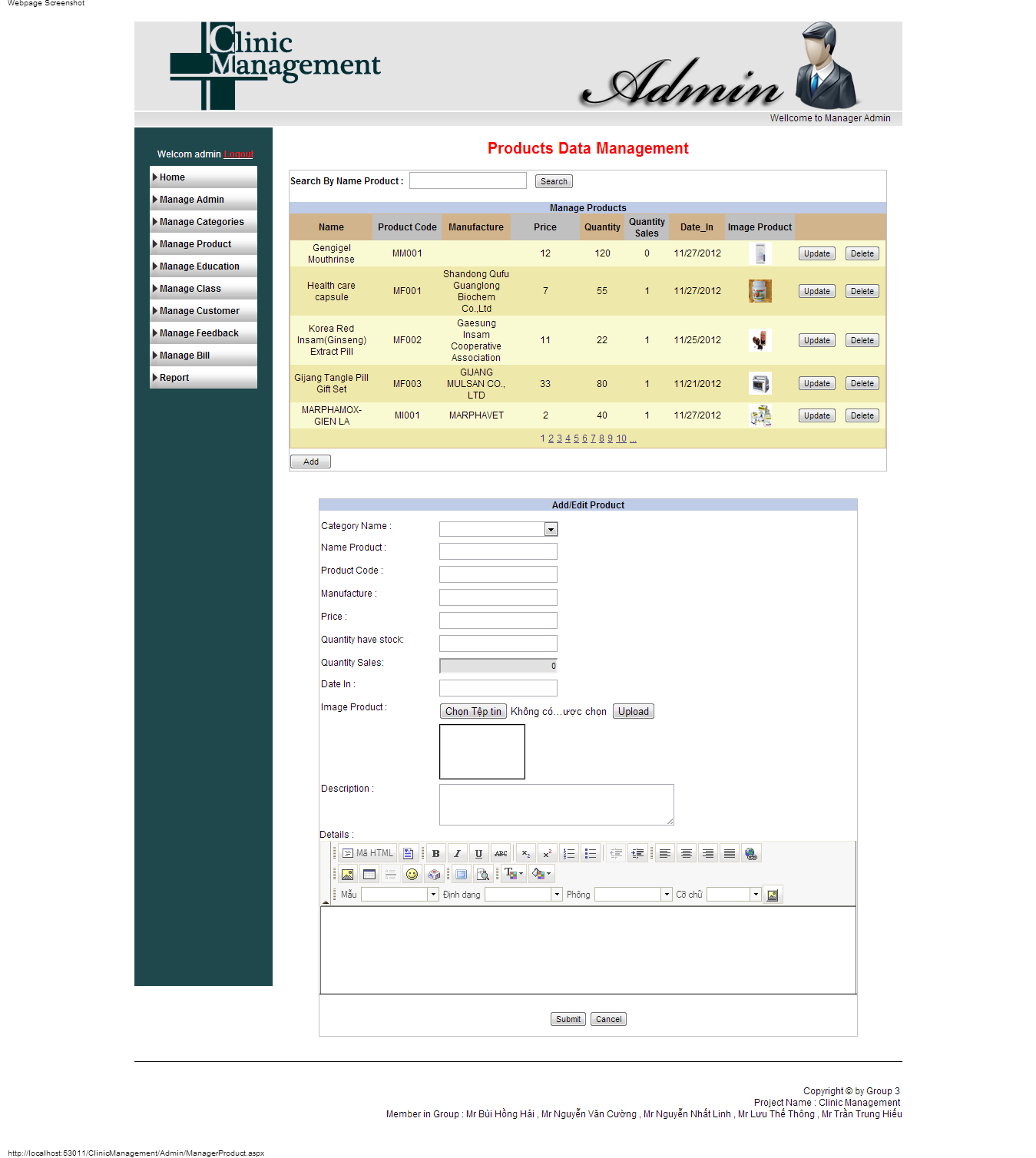
### Contact



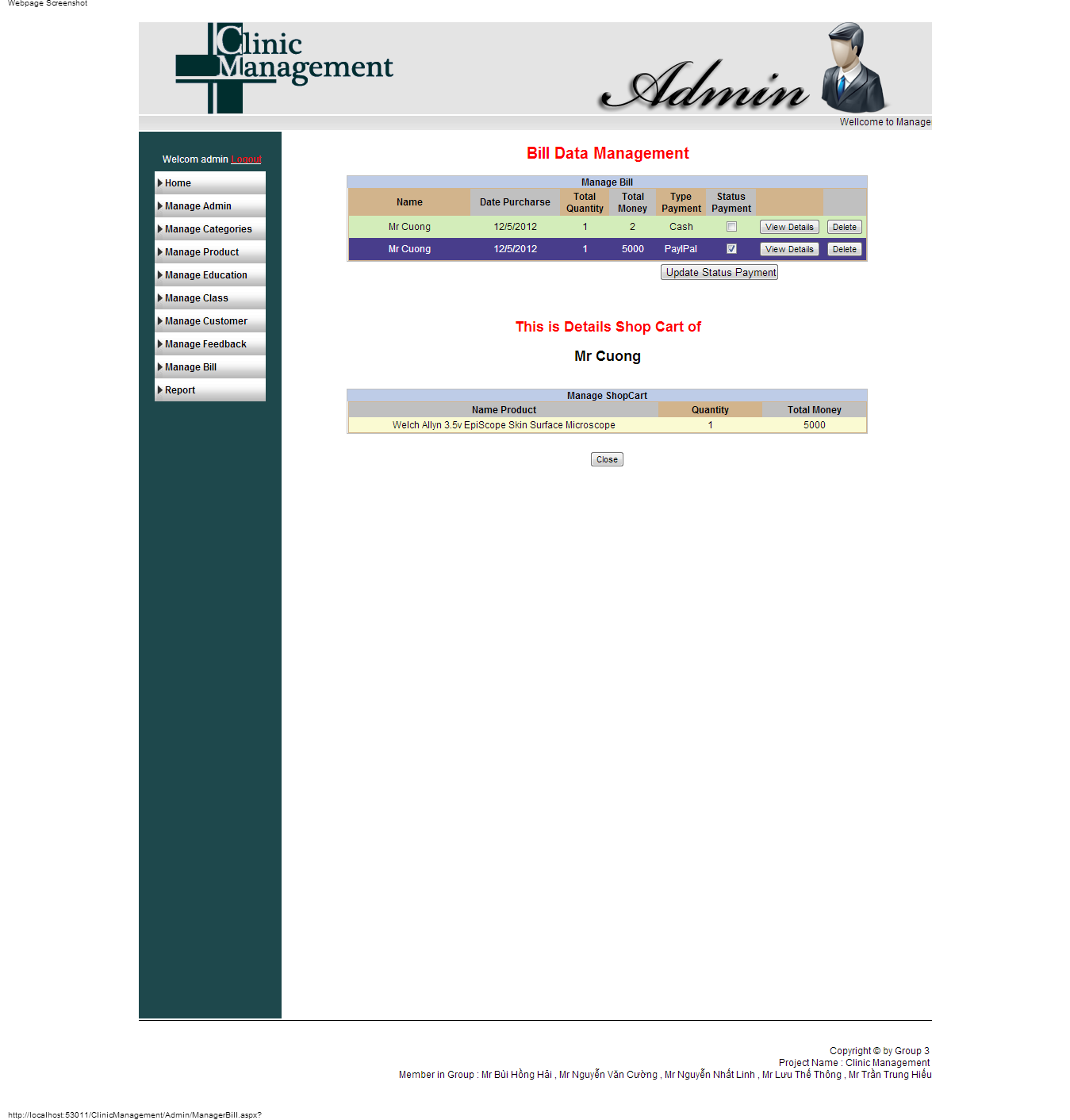
### Admin Home



### Manage Product



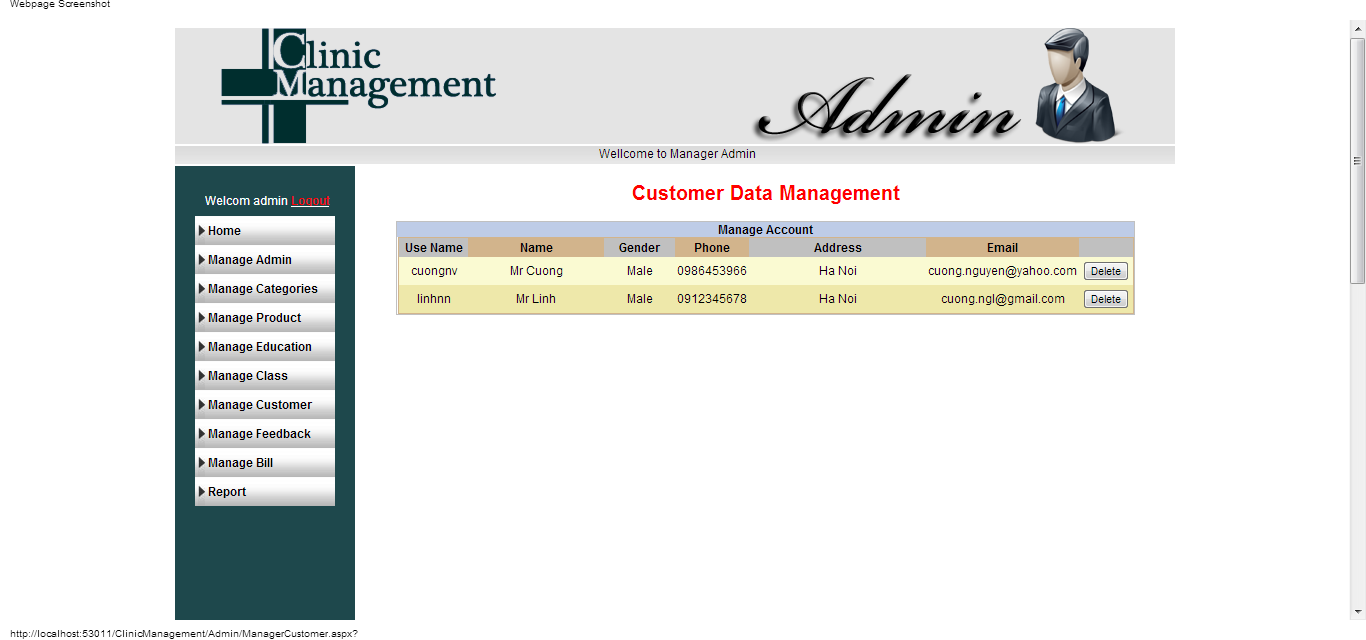
### Manage Bill



### Report



### Manage Customer



### Manager Class Register