

# Price-Drop Master: An E-shopping Price Drop Notifier

## Mini-Project Report

Hang Liu

h.liu.1@tue.nl

### I. INTRODUCTION

Nowadays, online shopping has already become an unseparated part of people's daily life. It provides an easier and faster way of satisfying the desire of buying things. This meets the demand of the metropolitan residents who need to live in the instant society today. At the same time, it also brings wider views and cheaper prices to the customers. The intention of this application is to provide a way of receiving personalized discount information of an online shop at the first place.

On the other hand, from the development of this application, I want to learn several new techniques to fill my knowledge gap in developing web applications. In this project, a web application is developed, which includes both front-end (web page) and back-end (server program, database). The techniques involved in this project are Python, HTML5, SQLite database and Web2py framework.

### II. PROBLEM ANALYSIS

#### *A. Problem Description*

Instead of aiming at a local market like a hypostatic store, web shops are facing to a much wider customer group, both in time and geography. This will cause a problem for the customer when the supply falls short of the demand. Such situation will happen when discount season comes. Good deals are often sold out within the first day.

Imagine the following case, you have waited for one month for the 'iPad Air' to have a discount. You checked its online price every day. However, at some point, you had a vacation to the beach. You enjoyed the sun so much that you forgot about your iPad. Then when you came back, your friend told you that he bought an iPad at the discount price, but this promotion had already finished. Will you be so regretted of missing this opportunity? The idea of this project is to provide an easy solution for this problem.

#### *B. Domain Analysis*

My first thought is to build a server application for automatic tracking of the product price. The server will do the job, so that the users do not need to check the price frequently.

With this idea, I made the following domain model, as shown in Figure 1. An application like this should contain a graphical user interface for the user operations. Then the user interface is connected to a back-end server, which provides the functionality to the user. In my domain model, the user interface can be either a smart phone application or a web page on a web browser. Both of them can communicate with the back-end server through HTTP request. On the server side, the application is responsible for handling the user request. The cron job is running by the server at a specific time frame, for example, doing regular price tracking task. The database can be accessed by both the application and the cron job. It is used to save user data.

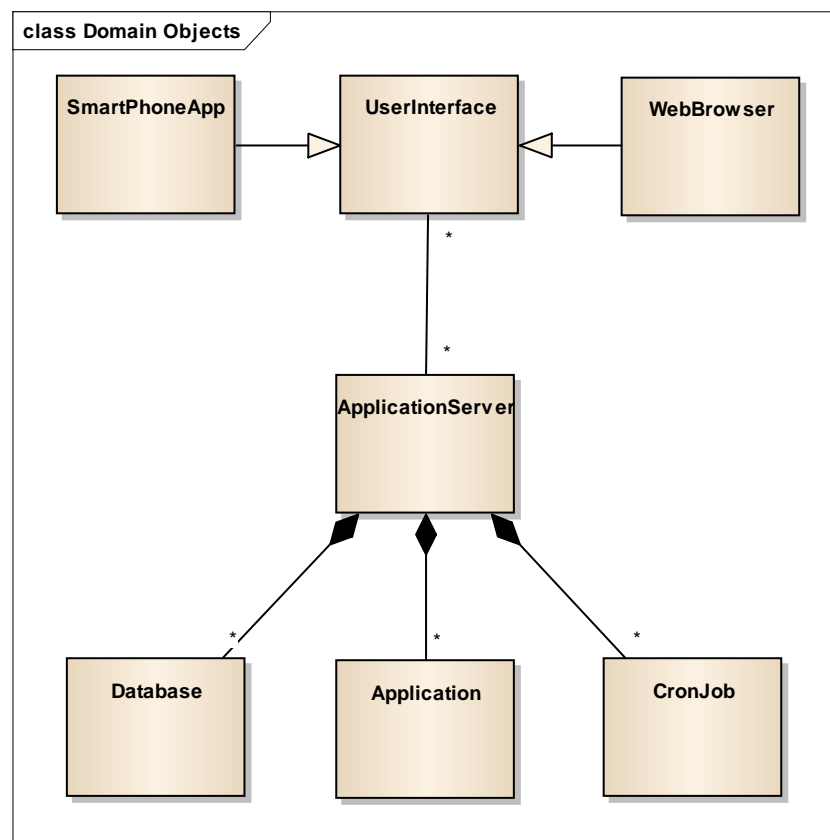


Figure 1 Domain model of a server application

### C. Use Case

Two basic use cases should be satisfied for solving the identified problem. First, the user should be able to set up a price drop notification. With this use case, the user specifies which product he wants to track. Second, the user should be able to manage his existing notifications. After a notification is not needed anymore, the user should be able to delete it. The detailed scenarios of these use cases will be introduced in the system design chapter.

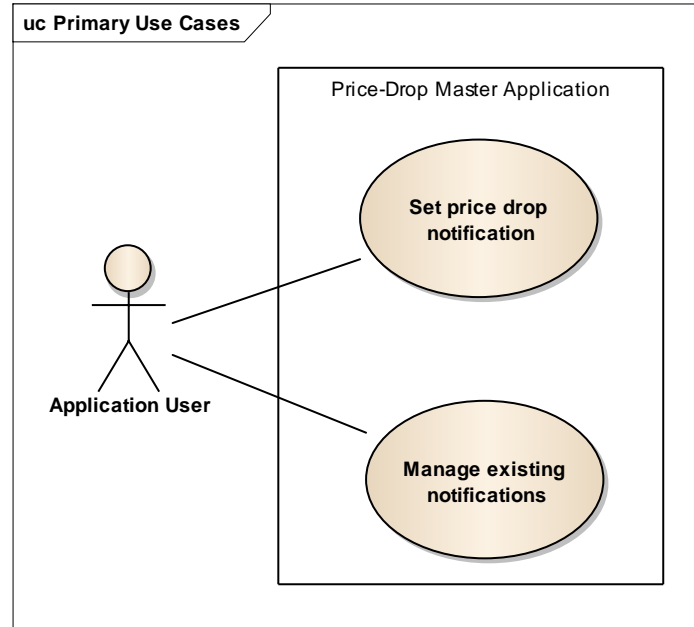


Figure 2 Basic use cases of the Price-Drop Master application

### III. SYSTEM ARCHITECTURE

#### A. *Technology Decision*

##### a) **User Interface**

As mentioned in the previous chapter, two options were considered for the user interface. Because the time constraint, only web-based user interface was implemented. It was chosen because of three reasons. Firstly, it is more common for people to do online shopping from their computer than from their smart phones. It will be easier for people to copy a web shop link into another web application than to insert it into their phones. Secondly, a web-based user interface also can be accessed from a smart phone. Although it cannot send app notification, an e-mail notification is enough for the purpose of this application. Thirdly, the HTML5 technique is new for me. I am more willing to learn a new technique than using a known technique in this kind of project.

##### b) **Web Framework**

Since I am also interested in learning the Python language, I was looking for a web framework that supports Python. After a short investigation, I decided to use the Web2py framework. As said on its website, it is a free open source full-stack framework for rapid development of database-driven web-based applications.

The reason to choose Web2py is very clear. First of all, it is very easy to start with. It only takes a few minutes to setup everything and run the hello world program. This is very important, since I need to learn and implement within 7 days. Second, it provides all the features that I needed. Besides the web server and the SQL database, it also provides a build-in cron mechanism. This cron task is platform-independent, so it also can be run on a windows machine. Third, the Web2py framework comes with a full set of documentations. Also it has a healthy community to support questions.

### c) Web Scrapping Technique

Web scraping is the main feature of the Price-Drop Master application. The initial choice is to use Scrapy. It is a fast high-level screen scraping and web crawling framework. However, after a short investigation, I found out that Scrapy does not fit this situation. Because Scrapy is built on top of the Twisted asynchronous networking library, so it needs to be run inside a Twisted reactor. The Twisted reactor needs to be created in the main thread. However, with Web2py framework, the main thread is used to build up the server. Other user applications are not running on the main thread. Since I want to keep using the Web2py framework, the Scrapy needs to be replaced.

Then I found the lxml library. It is an extensive library written for parsing XML and HTML documents. It uses XPath to locate information in structured HTML and XML document. XPath is a very simple API, which helps me to construct my own scraper for this application.

### B. System Overview

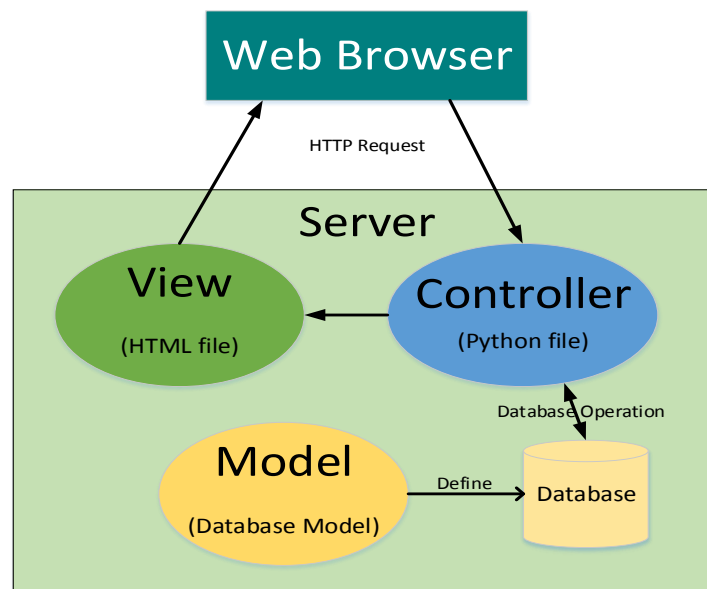


Figure 3 System architecture of a Web2py web application

The Web2py framework applies the Model-View-Controller architecture pattern. It separates a user application into three parts: data representation (the model), data presentation (the view) and the application workflow (the controller). The system architecture of a web application developed with Web2py is shown in Figure 3.

When the web user try to load a Web2py web application, the browser will send HTTP request to the server. The server first maps the requested URL to a function call in the Controller. The Controller may query/insert data from/to the Database. Then the Controller generates the output, which will be rendered by the View into a HTML page.

#### IV. SYSTEM DESIGN

The previous chapter introduces the general architecture of the Price-Drop Master application. This chapter will focus on describing the detail design of this application using the 4+1 view approach.

##### A. Scenario

This section describes the detailed scenarios of the two basic use cases.

<b>Use Case 1</b>
<b>Name:</b> Set price drop notification
<b>Context of use:</b> Online shopping
<b>Scope:</b> PriceDropMaster web application
<b>Level:</b> User-goal
<b>Primary Actor:</b> Web user
<b>Stakeholders &amp; Interests:</b> TBW
<b>Precondition:</b> PriceDropMaster web application is running correctly.
<b>Minimal Guarantees:</b> User's web browser is running correctly.
<b>Success Guarantees:</b> Target item is added into personal interest list.
<b>Trigger:</b> TBW
<b>Main Success Scenario</b> <ol style="list-style-type: none"> <li>1) Web user: Open the PriceDropMaster application through a web browser.</li> <li>2) Application: Present the main page to the user.</li> <li>3) Web user: Insert the web link of the target item.</li> <li>4) Web user: Insert an e-mail address, where he wants to receive notification e-mails.</li> <li>5) Web user: Submit the information.</li> <li>6) Application: Scrap the current price information of the target item.</li> </ol>

7) Application: If it succeed in getting the price, show the current price to the user and store these information in the system.
8) Application: Send a notification e-mail about the above subscription activity to the user.
<b>Extensions</b>
7a) Application: If it does not succeed, return a failure page to the user.

<b>Use Case 2</b>
<b>Name:</b> Manage existing notifications
<b>Context of use:</b> Personal subscription management
<b>Scope:</b> PriceDropMaster web application
<b>Level:</b> User-goal
<b>Primary Actor:</b> Web user
<b>Stakeholders &amp; Interests:</b> TBW
<b>Precondition:</b> PriceDropMaster web application is running correctly.
<b>Minimal Guarantees:</b> User's web browser is running correctly.
<b>Success Guarantees:</b> Personal subscriptions are managed.
<b>Trigger:</b> TBW
<b>Main Success Scenario</b>
1) Web user: Open the subscription management page, the link is included in the notification email from step 8 of use case 1.
2) Application: List the existing subscriptions of this user.
3) Web user: Delete an existing subscription.
4) Application: Save user operations to the system.
<b>Extensions</b>
3a) Web user: Change the notification e-mail address of an existing subscription.

## ***B. Process View***

Due to the time limit, only the first use case was implemented. Therefore, only the process view of the first use case is shown here. The sequence diagram of Use Case 1 is shown in Figure 4.

In order to make the diagram more readable, the function call to the Database Abstraction Layer (a class which will be introduced in the Logical View) is made directly to the database.

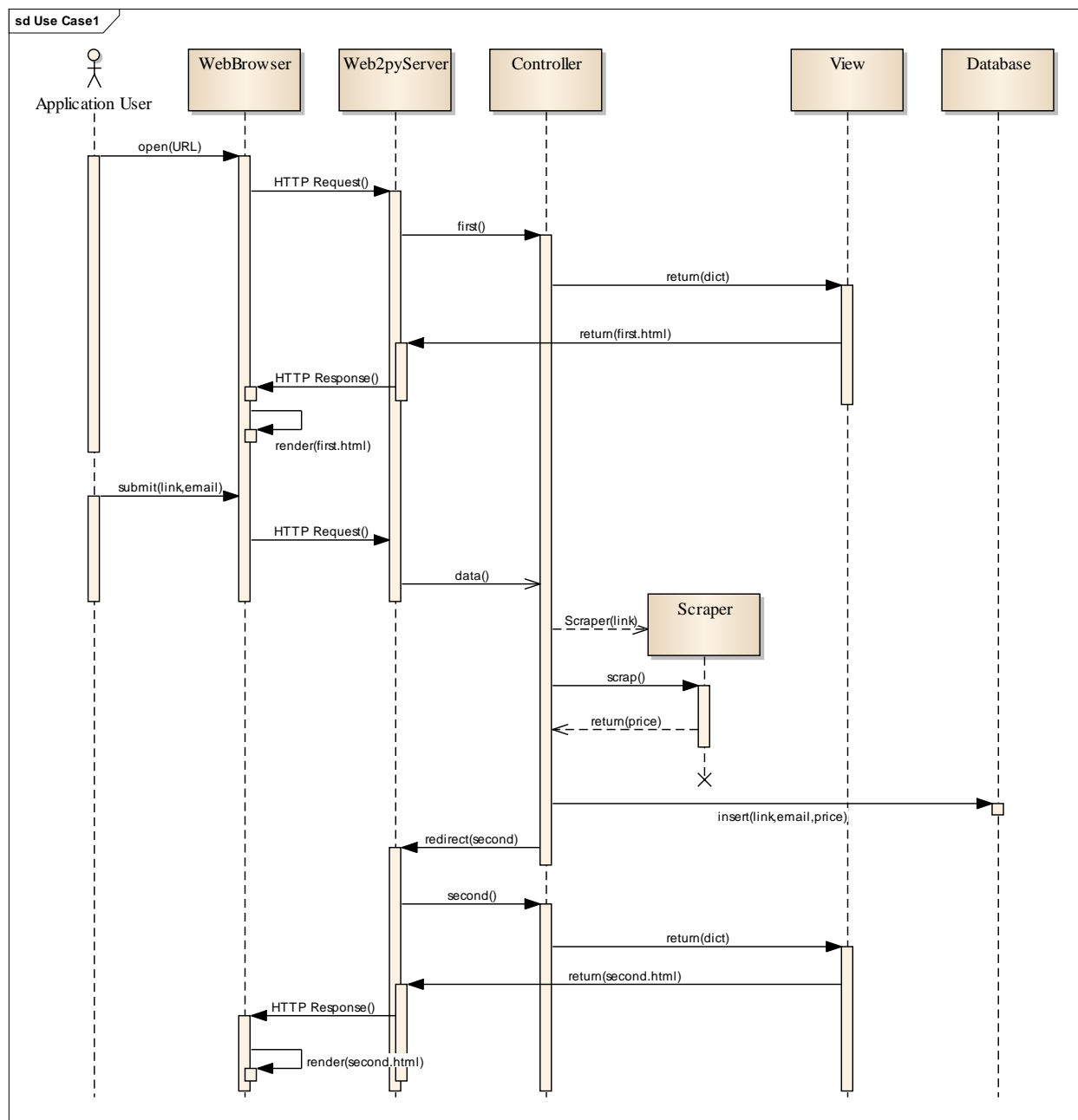


Figure 4 Sequence diagram of Use Case 1

### C. Logical View

The core part of Price-Drop Master application is the Scraper class. It provides the scrap() function, which is used to extract price information from a given website. This class is invoked in the Cron class and the Controller class. The Cron class defines a task which is executed at a defined time interval. This task is

doing the price checking job in the background. The Controller class defines the function that response to the user HTTP request. The return value of the function will be rendered into a corresponding HTML page. Then, all the data accesses to the database are managed by the Database Abstraction Layer class, which is provided by the Web2py framework. The class diagram of the Price-Drop Master application is shown in Figure 5.

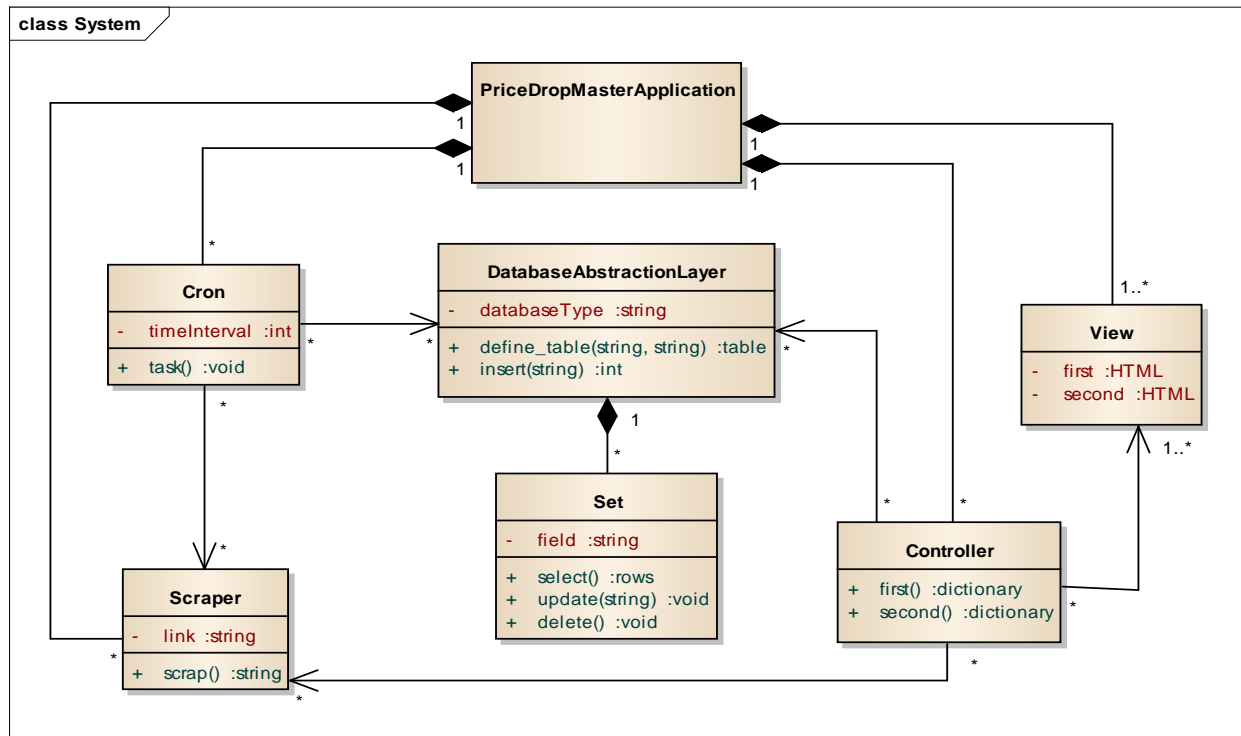


Figure 5 Class diagram of the Price-Drop Master application

#### D. Deployment View

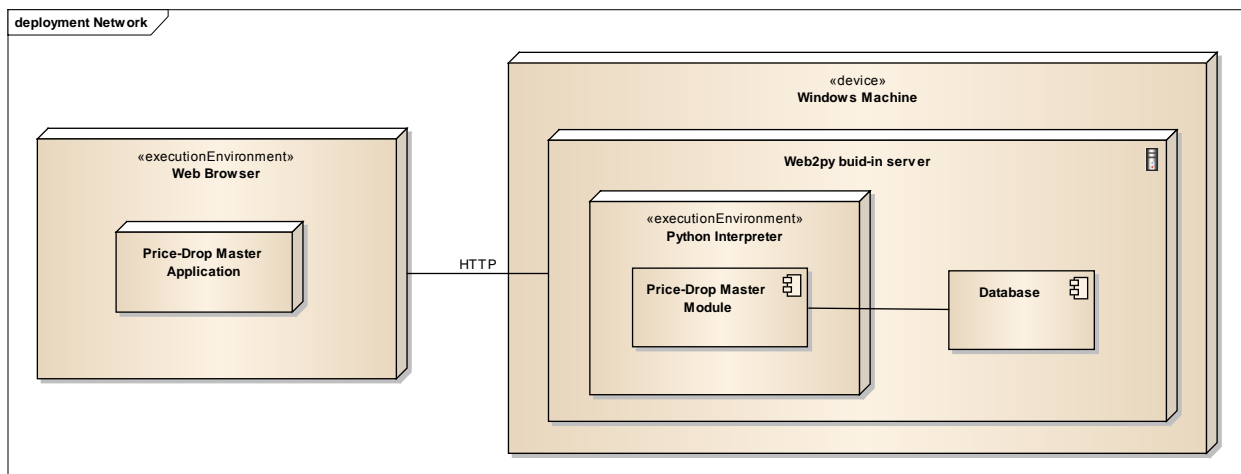


Figure 5 Deployment diagram of the Price-Drop Master application



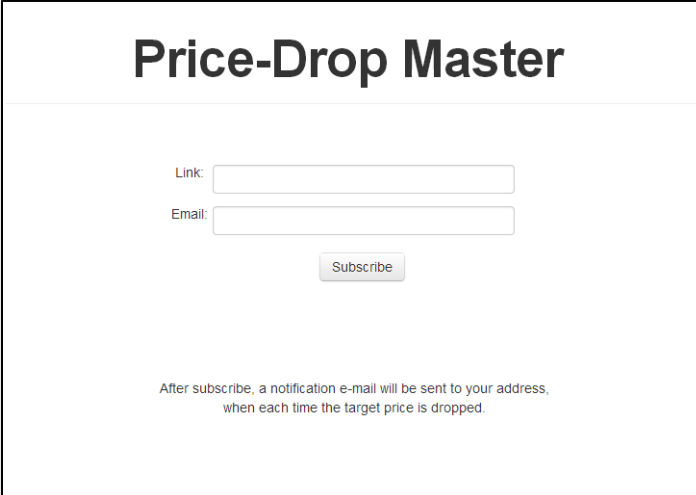
Figure 6 shows the deployment view of the Price-Drop Master application. The server is deployed on a single Windows machine using the Web2py framework. The server hosts the Price-Drop Master application together with its underneath database. The client can connect to this application using most web browser from any machines or devices.

### ***E. Development View***

Since the Price-Drop Master application is developed within the Web2py framework, the interfaces and ports among different components are encapsulated by the framework. For example, when the controller returns a dictionary, the framework will automatically invoke a HTML file in the view with the same name. These interfaces and ports are not exposed to external developers. In this situation, the development view will not make too much sense, so it is leave out in this report.

## **V. DEMONSTATION**

Figure 7 shows the main page of the Price-Drop Master application. The idea is to provide a simple interface to the user. User only needs to copy & paste a web store item URL into the Link field, and insert his e-mail address, where he wants to receive price drop notifications.



**Figure 7 Main page of the Price-Drop Master application**

After subscribe, if the subscription is success, the user will receive an e-mail in his address that confirms with him about the subscription. Also the web page will jump to the success page as shown in Figure 8.

However, if the subscription is not success, the most probable reason could be the web store inserted is not supported by this application yet. In this situation, the web page will jump to the failure page as shown in Figure 9.

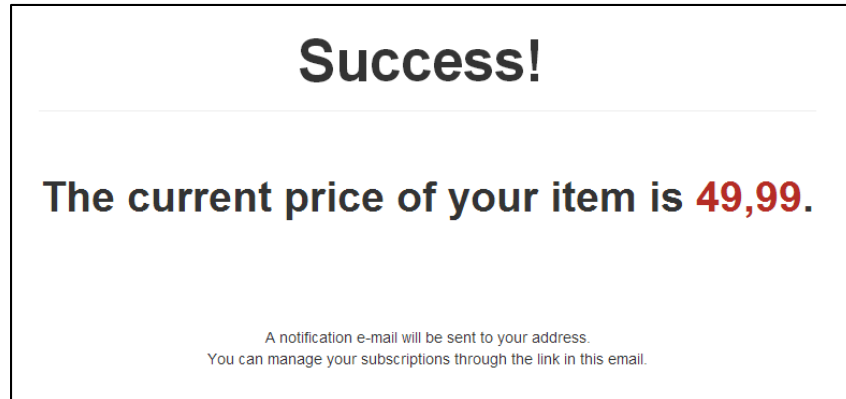


Figure 8 Success page of the Price-Drop Master application

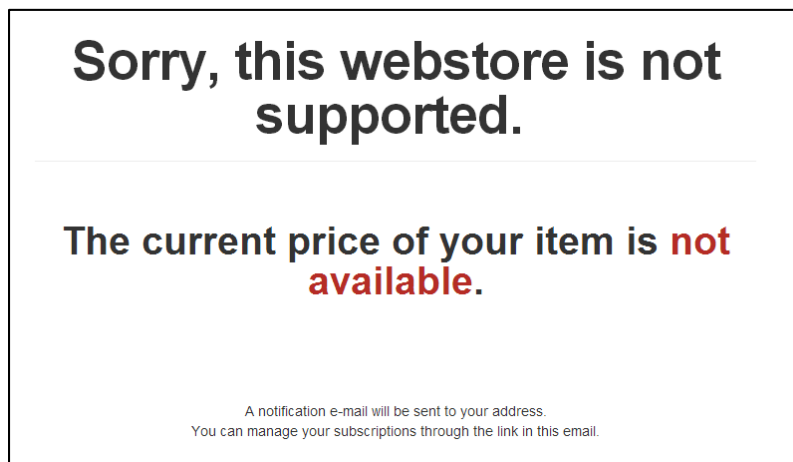


Figure 9 Failure page of the Price-Drop Master application

## VI. CONCLUSION

This report summarized the works that have been done in this mini-project. This project is mainly focused on the learning part instead of developing a product. Four major techniques were experienced during this project, Python, HTML5, Web2py framework and web scraping. This project was started with exploring the web application domain and investigating the available frameworks. As a result, the Web2py framework was chosen due to its fast learning curve and its ample functionality. Then some problems were experienced when investigating the web scraping technique. At the end, the lxml library was used for handling the scraping task.

The main use case of this application is implemented. There is not enough time for finishing the second use case, although there is no technical challenges left.

Future extension to this application is also possible. This application can be extended to monitoring all different types of website change, for example monitoring the exchange rate between different currencies.