# Chapter 3 Product Development (3901 words)

## 3.I An introduction to Product Development

Product development is the process of coming up with an idea for a potential product and taking it all the way to completion. This can be a new product or an upgrade/extension of an existing product, either way it will have to follow a development process for the product to be created efficiently. There are different methods for product development but they all basically break down a large task into smaller steps making the whole process more manageable. One method is known as the Waterfall method and that is what we used to develop this product. In the Waterfall method you come up with a set of requirements, then you start planning the solution, once the product is planned out you create it, and finally you test it to make sure it works. There are 2 large drawbacks to this method; if the requirements are not clear then the product may not solve the issue and secondly, any changes to the plan during implementation or testing will prove very expensive and take time.

## 3.II The Team Product

### 3.II.a The product specification

Our product is aimed at Hotel owners in general. It serves as a helpful tool to manage review sites, such as TripAdvisor. The GUI side of the program is in the form of a website, utilising HTML and PHP.

This side of the system should allow a user to log in or register and use the system. There should be a project management pane in the settings that allows a user to manage their project on the system. This allows a user to change the URL of their hotel on TripAdvisor. There should be a general settings pane that allows for other utilities such as password changing.

The project tab on the site is where all the scraped reviews will be shown. This page will allow the user to look through reviews that have been pulled from the trip advisor site, which will include the text from the reviews and the overall ratings per review. There will be features that allow users to view these in a graphical format, by using interactive graphs and charts. The user should be able to map their hotel average review rating by time in a graphical format which will allow the Hotel Owner/Manager to keep track of months that they did well in, and months they didn’t. The reviews will be sorted by time so they should be able to read up on the reviews that got them better ratings that month.

To do all of this, the scraping element will be coded using Python, and extensions such as Beautiful Soup, Requests and pymysql. These will allow for the easiest and most efficient scraping method and storage of the data, especially when it is used with the website, as were summarised by Ben after they were discussed in the CE101 Lectures [1]. Simas further went on to describe the Requests module which gave for a better understanding on how the module works. This allowed us to plan the usage of it. [2] That will be coded using HTML, PHP, JavaScript elements and SQL. Both the sides of the code will communicate with a MYSQL database.

The database will store all the data from the system. This will include some global settings, scraped reviews, projects and user data.

System will be hosted in virtual private server, so it could be easy accessed via internet connection. There will be modules and clients installed to be able to run PHP, Python and MYSQL. The user of the product will not be asked to install any specific programs and the wide range of internet browsers will be supported.

### 3.II.b The product design

The project needed a lot of time dedicated to designing the system so that both sides (Python and PHP) would work in harmony.

The project design starts with the database. To create the system we would need a database hold all of the information from TripAdvisor and for user logins. The database has to be fully functional and normalised to avoid redundant data. It would need many tables: Owner (to hold all of the information about the users); Project (to hold information about the each users project setup, such as the hotel URL); ScrapedReview (to hold all the information scraped from the physical webpages); Session (to hold information about the project status, for example, the last time it was reviewed) and finally SessionhasProject (to physically link the session and project and to keep the database normalised). A database schema was drawn up by Simas for the basic database needs before work began creating and implementing this. [4]

All of the data scraping, recording and validation will be processed on the Python side of the system. When the user enters their details and all information is validated in the GUI, a new session should be created and the Python script will be invoked. The session id, project id and the project URL should be carried over from the GUI side to the Python. At the beginning of the process, Python should scan the settings table located in the main database to adjust its configuration. The admin should be able to define how many reviews are written to the database per connection, the waiting time for each connection to avoid overloading the hotel review site servers and the method that should be used for review text extraction.   
When the required configuration is deployed, the script will check if the website is online. If the check is successful the first scan will start to determine how many pages should be scanned. If it is not the first time project has been scanned, the program will check where it finished its scan last time, so server resources are not going to be wasted and duplicates will be avoided. The main scraping process will be placed in two loops. First loop will download the whole page and will convert it in to the BeautifulSoup object. This will allow us to parse the html page and make it more manageable. The second loop should extract HTML tags were the reviews are placed. When we will have the desired data, the source will be fed to defined functions for text, rating and date extraction. All the data will be stored using tuples, which then will be added to the main list. There should be some checks implemented to control loops, as too many inactive loops could crash the system or slow it down. When the program is initiated the timestamp should be set, which will allow us to break the loop if it takes too long to complete. When all scanning is complete and all the data has been transferred to the database, the script should close all connections and update the session so the GUI part will be able to determine that the scraping is done. If the scan should fail the appropriate status code will be recorded to the session row and the appropriate message should be displayed.

Several external Python modules will have to be used. Beautiful Soup will be installed to manipulate HTML source code. This module makes this process much easier as you can convert the source code to the object and navigate between the HTML tags. Second module will be pymysql, we need to install it because Python does not have built in support as PHP does. The Requests module will be imported, so we can establish HTTP connection and download the desired website.

The GUI side of the system, which is based around a website, will need to be a fully functional website but also interact with the python code to scrape the websites. The system must allow users to login and access their project. This will entail the users going through a login process and being directed to a home page where many session variables would be started to hold details about the user. If the user does not have a login they can create one, which should update the database upon registration. Once the user has logged in they should be able to setup a project which should allow the user to select the hotel of their choice, and also edit basic settings, such as passwords and also the hotel URL address, the system should constantly update the database with new settings or to check entered passwords and other elements.

When the user has logged in, they should be presented with different pages, in an attractive layout using tabs, and should be able to switch between pages, giving an introduction to the system, information about the licenses that we have had to use to create the system and information about the project and maybe some links to documents such as the finalised team report and design documents.

When the user selects the scrape tab, they should be given the choice to use old data, or refresh the database. This is when the PHP should interact with the python script, and begin to scrape the hotel review site for information. The user should be informed that the system is working, because it will not be instantaneous due to the delay in the queries to the review site. Once the python script has finished, the database will be up to date and the system should query this to get data on the users reviews, ratings and statistics from the review site and present this all in one neatly presented page. There should be graphical elements such as graphs and tables and also the plain text reviews all laid out in a tabular fashion. The user should be able to filter reviews by date and easily analyse their reviews by using the predefined elements on the page.

Once the user has finished looking at their reviews, they should be able to simply logout of the system by clicking on the logout button. This should destroy the session data on the system.

To do this we have chosen to use the Bootstrap 3 framework available on an open source license. This will allow for maximum compatibility across many devices and browsers, new or old. It will also allow the system to look modern and comes with many features such as pre-defined div elements, Glyphicons and JavaScript elements such as modals and interactive graphs and charts. Using this framework will allow for the system to look professional and modern with less time dedicated to the styling. This was summarised by Ben and given as a presentation to the group, he mentioned some of the features that would be utilised and gave some examples. [5]

The language of choice for the website will be PHP. This is a great language for creating dynamic websites as it is very versatile, secure and also is widely available across many webservers that we will have access to. There will be elements created using JavaScript and a html error page and index.

To make an efficient GUI, Jeffery took some time to produce a quick document that gave us an idea as to the layout for our GUI. To do this he took time to look at commonly used, popular review sites for Hotels, which were TripAdvisor and Booking.com and summarised what makes them a good and easy to use system. [6] To give us an idea of other Hotel Review sites and how they looked, Emmanuel provided a collection of screenshots from live sites [7] and Ted also provided layouts for other online sites that scraped data together, which helped with the design for the pages on the system that provided user output. [8][9]

The database system we are using will be PHP MYSQL. This will allow the utilities of MYSQL to be accessible on a webserver. The server that we host on will define the specific software that is used to manage the database, but we know that it will be local to the system, and will work with both PHP and Python.

A web server with a specific configuration will be required. The project will be hosted in virtual private server running an Ubuntu (Linux branch) operating system. This operating system has been chosen because it's free of charge, easy to configure and properly maintained. In this machine we will install and run NGINX (an alternative to Apache) as a HTTP server, so our project will be able to deliver the pages to the client and a MYSQL server to be able to store the reviews and customer data. It is important to pay good attention to the MYSQL configuration as it will be used heavily and there would be moderate amount of data stored inside. As well, we will have to install Python with all required modules. There are few Python module managers available, so it will make the process easier. To enable GUI we will have to install and configure PHP-fpm server. There would be additional configuration required to NGINX server, so it can use PHP-fpm server and deliver the PHP pages. In addition to that minimal server security will have to be ensured – we will have to enable and configure firewall, set up appropriate access levels to users and make sure that the operating system with all installed clients are updated regularly.

### 3.II.c The product implementation

Implementation of the product involved the purchase and setup of a small web server. This allowed us to host the product online, where it would be stored for general use. This is a better solution than having a local based program as we would not need to install any modules to run the system.

The code itself was created in a procedural fashion, because both of the programmers were more familiar with this layout for code. The PHP could have been coded in a procedural fashion, however this would have only worked best for the MYSQL queries, however they were laid out into pre-configured variables anyway. The python code worked best in a procedural layout, as it was easiest just to pass lists around, and not objects, due to the formatting each time a query would need to be executed.

There were many functions created for the Python side of the system, more than one for each job, as it was easier to share functions between an elements of the code. These functions included review\_text, long\_text, mysql\_session\_update and connect. These are very self-explanatory, but are explained in Simas’ list of functions document and the code. [10][11] For the PHP side of the system, there were a few functions created, such as, checkadmin, checkurl, checksession, pwcheck, pwhash and checklogin. These all were given logical names and their purpose commented in on the code. [12] The main body of ‘functions’ for the PHP was a large if loop. It utilised the form elements on the website, and checked for the form that was submitted against its given name in the POST data. This allowed for specific parts of the code to be executed.

Simas took control of the server setup, being very experienced with Linux, this was an apt job for him. A server from BalticVps, a very reliable provider located in Lithuania was chosen to host. For the beginning the basic type of Virtual Private Server (VPS) looked completely enough. We installed an open source operating system Ubuntu 14.04 64 bit edition. The system was upgraded and all required modules were installed via apt-get application manager.

Firstly Apache2 was installed so we can deliver the web pages. We used a default configuration, as there were no additional modifications needed. After that the PHP5 client and MySQL server was installed with all required modules. We had to have those modules in our server to be able to run GUI. The Apache2 has its own module managing service, so this helped us out a lot. When we had all the parts which were required for GUI, the Python3.4 was installed so we would be able to run the scrapping script. To save some time, the PIP was installed as well. This small application lets us to add most popular Python modules much easier. With help of PIP all required modules were installed. There were some changes made to the default firewall so the scraper can be run smoothly.

The server was documented by Simas, and a small document was created telling our team how to get onto the server and view the files. [13]

Once the server had been setup, to use the system, the end client has to just point their browser URL to the IP or registered domain. The one we registered was a .tk domain, [www.hotelscraper.tk](http://www.hotelscraper.tk) [14] [15].

Everything worked well with the system except for the database connections. This was the only code we had to change when moving from local machine hosts to an online one. We had to change the login details for the MYSQL database to match the one we had setup on our server. You can see this change at the top of the PHP constants page, where it lays out the connection details in a small variable. [16]

### 3.II.d The product testing

The system needed constant testing whilst being created. This fell under general debugging. Python allows the use of IDLE, which will dynamically check the code whist it is being interpreted. This allowed for great help when creating the Python side of the system. However it was not such help when using the Python modules that were imported and a lot of time was spent analysing and learning how to use the specific modules themselves.

There is no syntax checker that comes stock with PHP, so that was created without one, but being experienced with PHP helped. There were many bugs on the way with the website side of the system, but all were ironed out by debugging and testing as the development progressed. To be able to run the system, a local host server had to be setup, that was capable of processing PHP scripts and also had a PHP MYSQL database setup. phpMyAdmin was used for the database setup on the website side of the system whilst the development was ongoing. This was a helpful tool, as SQL queries could be tested and checked using the software.

Whilst coding the system, we used a form of agile development which entailed test-driven development within each cycle. This allowed us to create the system in smaller chunks. The code was split into 2 halves, the PHP and the Python. It allowed the programming specialists to focus one part separately. When developing, the code was updated in small cycles, and at each stage, the code was tested for bugs. It allowed for a more bug free product to be rolled out at the end of development, and helped when we came to join the systems together and display all of the scraped data we had formulated. For example, the cycle where the Settings pages on the website were developed, was done in one chunk, and that was split down between each element on the page. So the code would be created, tested until the developer thought it was bug free and uploaded. These were all tested at the end by another team specialist. So the individual development of functions was test driven, however the segmentation of sections was agile development.

Once the system was live, Desmond took the job of testing the system on multiple browsers. The first main bug that he found, was that the system would not work on Internet Explorer. This is a big bug, but I believe with some configuration it would work. [17] When developing it would work on the local machine host, so it may be a server issue. However with other browsers it worked well. All fields were tested, Desmond tried to enter incorrect values in fields. He documented all of the errors that he found. During his testing, he found a few minor errors, however most of the system worked the way that it should. The first error he documented [18] was that when a non TripAdvisor URL was inserted in the settings page, the URL was not accepted, but an error was not displayed [19] shows a bug that when the user tries to change their password, if the new password does not match the confirmation field, the password is not changed, but no error message is shown. These were the only bugs found in the system. There may be a few, but most are very minor and do not affect the system at all, except from displaying error messages. The big bug was that the system would not work with Internet Explorer, which was bad news because the whole idea of using the Bootstrap framework was that the html would work on any browser.

## 

## 3.III Context

### 3.III.a Legal matters

When we think of legal matters, instantly the principle of ‘intellectual property’ is introduced. Here, we have many rules to abide by in order to successfully develop a product, one of which is **copyright**, this is the authenticity of the actual idea behind a product. In our case, as far as we were aware we had thought of an entirely unique program which is not based on any prior creation so we were not breaching any terms or conditions. Furthermore there was the issue of **patentability**, this is where the questions ‘is the invention/creation new?’ and ‘what sets it apart from the rest?’ come into play, the simple answer was yes because as opposed to creating merely another single hotel review site, we were using multiple to get broader reviews and comments.

To aid us develop our product we used further programs such as Google charts and bootstrap, Google Charts helped us with visualizing data on our website, it provided several ready-made chart types.[20] In order to use this we were given the responsibility of protecting the privacy of users (Those who submitted the hotel reviews) as our data manipulation was based on their feedback. Another program we used to help with the aspect of HTML was BootStrap, in order to use this however we needed to provide a purple and white trademark logo. [21]

### 3.III.b Ethical matters

Ethics is defined as a ‘branch of philosophy that defines what is good for the individual and for society’, applying this to our project the only possible setback we may have had in terms of ethics was the lack of consent given by the actual reviewer, as we were using many reviews from different sites. Asides from that, there were no obvious ethical issues in our program which would have severely affect the production.

### 3.III.c Health & safety matters

Health and safety is explained as **ALARP** (As Low As Reasonably Practicable) and SFAIRP (So Far As Is Reasonably Practicable). ALARP is a form of reducing risk but in our case, our project was based around a hotel-review sites so there was no direct danger imposed on any user of the program. **SFAIRP** is what we actually mean by ‘reasonably practicable’, it involves a healthy balance between the ‘quantum of risk’ vs ‘the sacrifice involved in averting risk’, if there is gross disproportion then the project as a whole is under threat but as expressed before, there was no real hazard caused by our project so the effort put into reducing hazard was minimal and this mirrored the size of the potential actual risk/hazard/danger.

References:

1. CE101 lecture – Week 6 lecture from CE101 course – Anthony Vickers

(<https://moodle.essex.ac.uk/mod/book/view.php?id=208150&chapterid=1893>)

1. Extension summaries from w6 lecture.docx – Description of python extensions from CE101 Lecture – Ben Higgs

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/88347/extension%20summaries%20from%20w6%20lecture.docx>)

1. requests\_precis.pdf – Precis of how to use Python Requests module – Simas Kairys

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/89390/requests_precis.pdf>)

1. Mysql.png – Image of first proposed Database Schema – Simas Kairys

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/96791/mysql.png>)

1. Bootstrap pres ben.pptx - Bootstrap3 summary – Ben Higgs

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/90992/bootstrap%20pres%20ben%20.pptx>)

1. Review website research – Hotel Review site analysis – Jeffrey Chujor

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/89928/Review%20website%20research.docx>)

1. Website Design.docx – suggestions for web design - Emmanuel Odame-Asante

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/88295/Website%20design%20.docx>)

1. Allkeyshop URL and screenshot – Ted Fey

(<http://www.allkeyshop.com/blog/> , <https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/88065/allkeyshop.jpg>)

1. GoogleShopping URL and screenshot – Ted Fey

(<https://www.google.co.uk/shopping> , <https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/88069/googleshopping.jpg>)

1. Python code final.zip – Zip of finished python code – Simas Kairys

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/97954/PYTHON%20code%20final.zip>)

1. List.odt - List of functions in the Python code – Simas Kairys

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/97942/list.odt>)

1. PHP code final.zip – zip of finished PHP code- specific referenced documents inside (functions.php and constants.php) – Ben Higgs

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/97950/PHP%20code%20final.zip>)

1. Accessing\_server\_pdf.pdf - Document explaining how to access the web server – Simas Kairys

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/95727/accessing_server_pdf.pdf>)

1. Hotelscraper domain link – Ben Higgs - <http://www.hotelscraper.tk/>
2. IP link to the server (no DNS) – Ben Higgs - <http://185.69.53.182/login.php>
3. PHP code final.zip – zip of finished PHP code- specific referenced document inside (constants.php) – Ben Higgs

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/97950/PHP%20code%20final.zip>)

1. CE101 bug test.docx - CE101 bug test page 16 – Desmond Wei Chua

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/96707/CE101%20Bug%20Test.docx>)

1. CE101 bug test.docx - CE101 bug test page 9 part 4)a – Desmond Wei Chua

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/96707/CE101%20Bug%20Test.docx>)

1. CE101 bug test.docx - CE101 bug test page 13 part 4)e – Desmond Wei Chua

(<https://moodle.essex.ac.uk/pluginfile.php/323269/mod_data/content/96707/CE101%20Bug%20Test.docx>)

1. Google charts (<https://developers.google.com/chart/interactive/docs/security_privacy>)
2. Bootstrap (<http://getbootstrap.com/about/> )