2016

**Report**

**For teaching set09103**

**Jacint Virag - 40270622**

Contents

[Introduction 2](#_Toc468279010)

[Home page 2](#_Toc468279011)

[Product page 3](#_Toc468279012)

[Shopping cart page 3](#_Toc468279013)

[Register page 4](#_Toc468279014)

[Search page 4](#_Toc468279015)

[Admin Panel 4](#_Toc468279016)

[Admin Add page 5](#_Toc468279017)

[Design and architecture 6](#_Toc468279018)

[Folder structure 6](#_Toc468279019)

[URL structure 6](#_Toc468279020)

[Enhancement 6](#_Toc468279021)

[Database 6](#_Toc468279022)

[Front-end and functionalities 6](#_Toc468279023)

[Shopping Cart 7](#_Toc468279024)

[Critical evaluation 7](#_Toc468279025)

[Personal evaluation 7](#_Toc468279026)

[Resources 7](#_Toc468279027)

[Frameworks 7](#_Toc468279028)

[Content 8](#_Toc468279029)

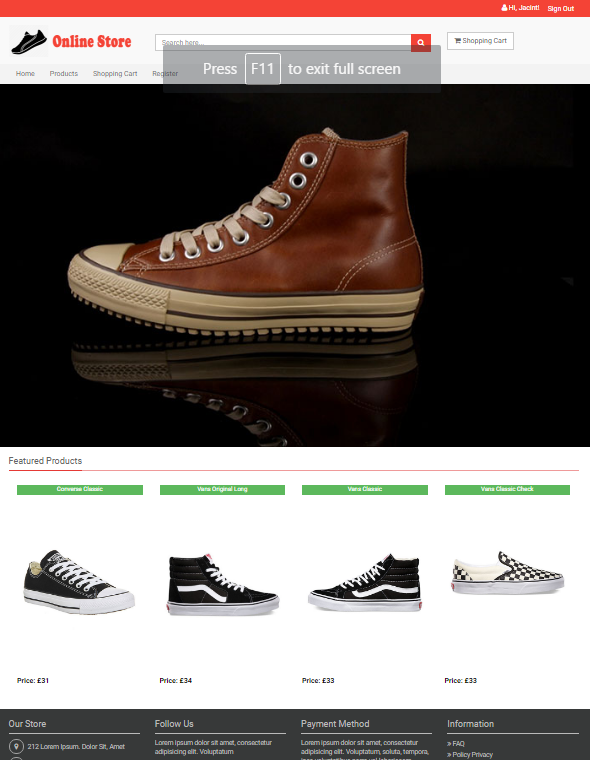
[References 8](#_Toc468279030)

# Introduction

The web-app I developed is an online shop where the users can browse through products from different type of shoes. The user can register, add items to the shopping cart and search within the products by name or description. The user can log in as an admin and able to modify or delete products and its metadata and able to add new items into the database with file upload.

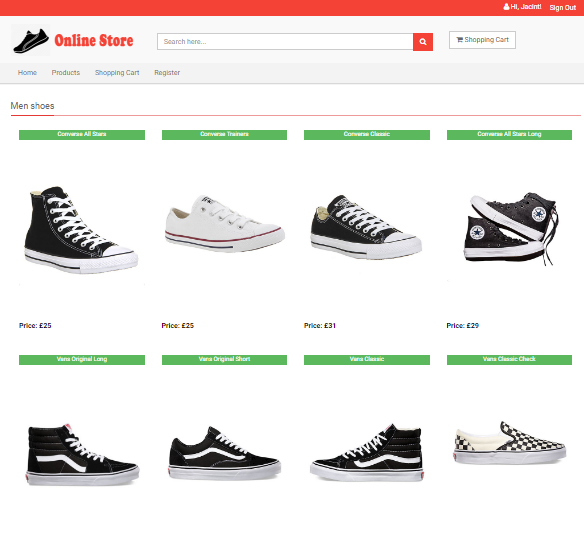
## Home page

The home page includes a slider, search bar, login form and a list of featured items.



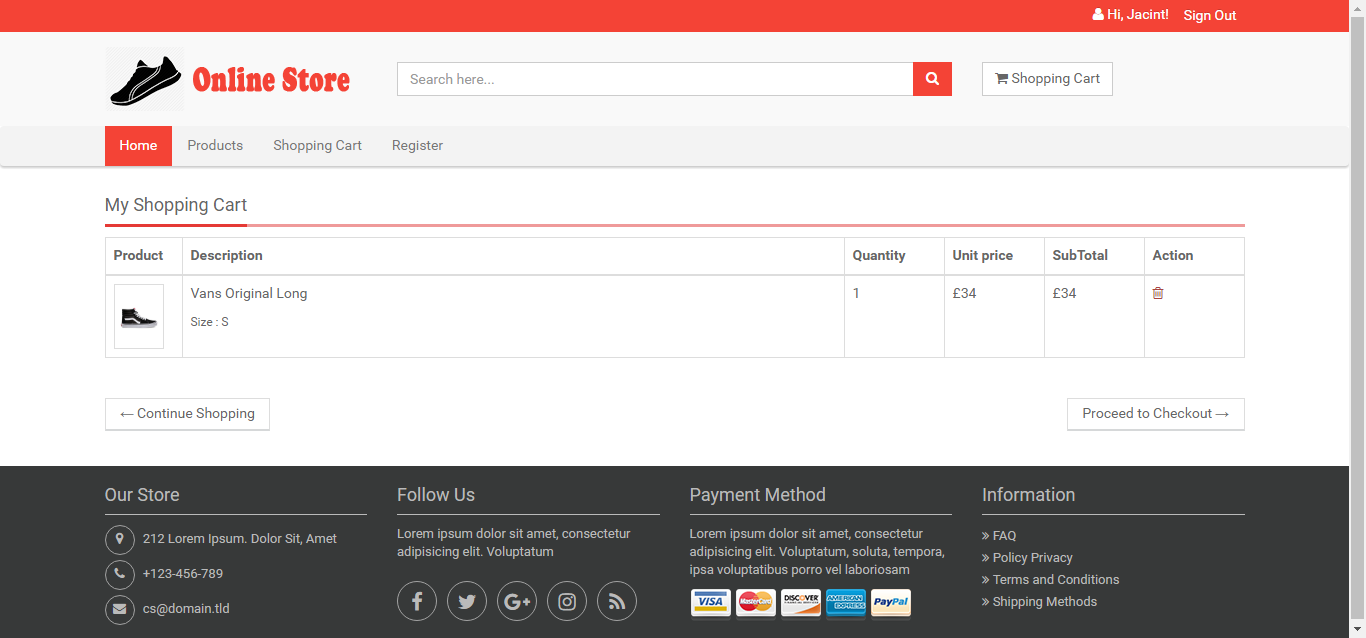
## Product page

This page lists all the products from the database.



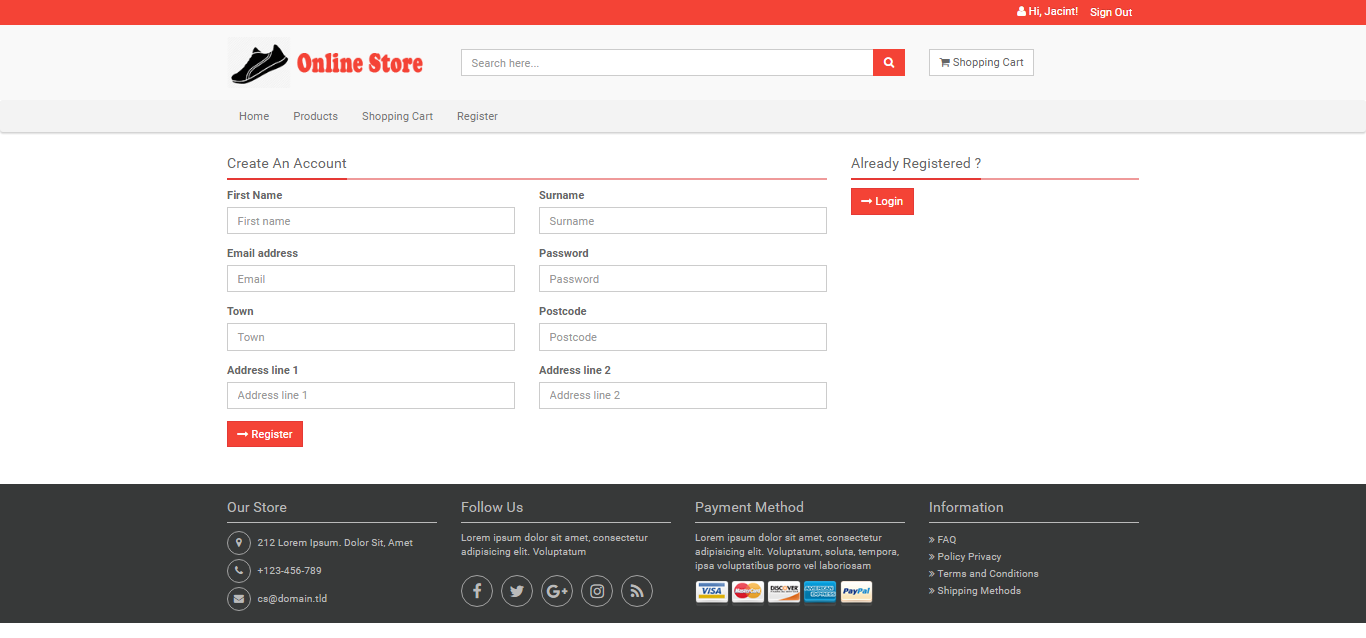
## Shopping cart page

This page lists all the items added to the shopping cart. The user can delete unwanted product from the cart.



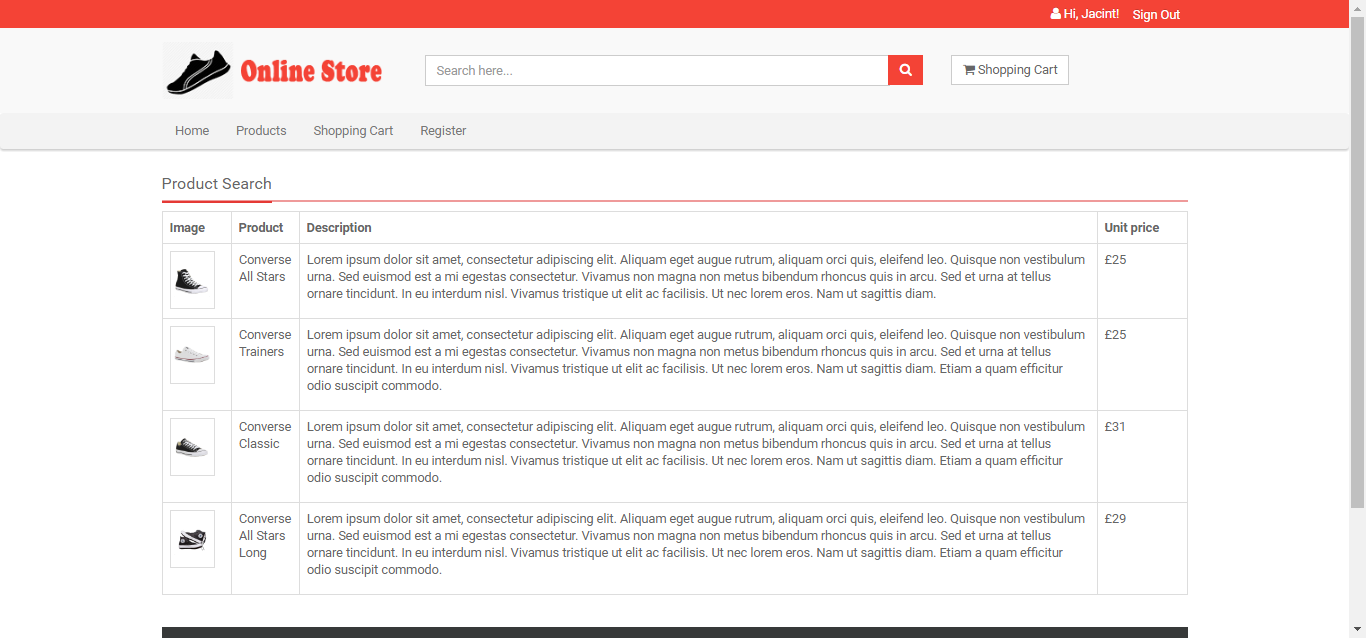
## Register page

This page enables the user to register. The form checks if the username is taken if so it comes back with an error message.



## Search page

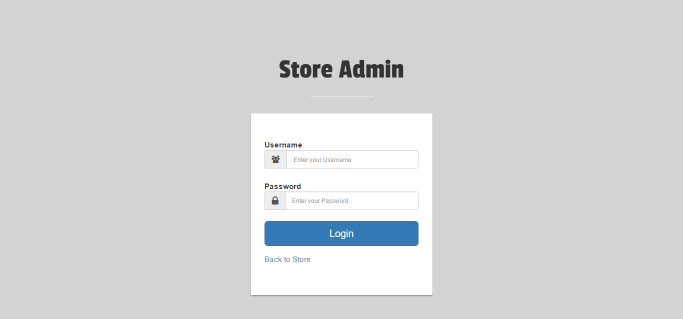
The search bar is constant on every page so the user can freely make a search any time.



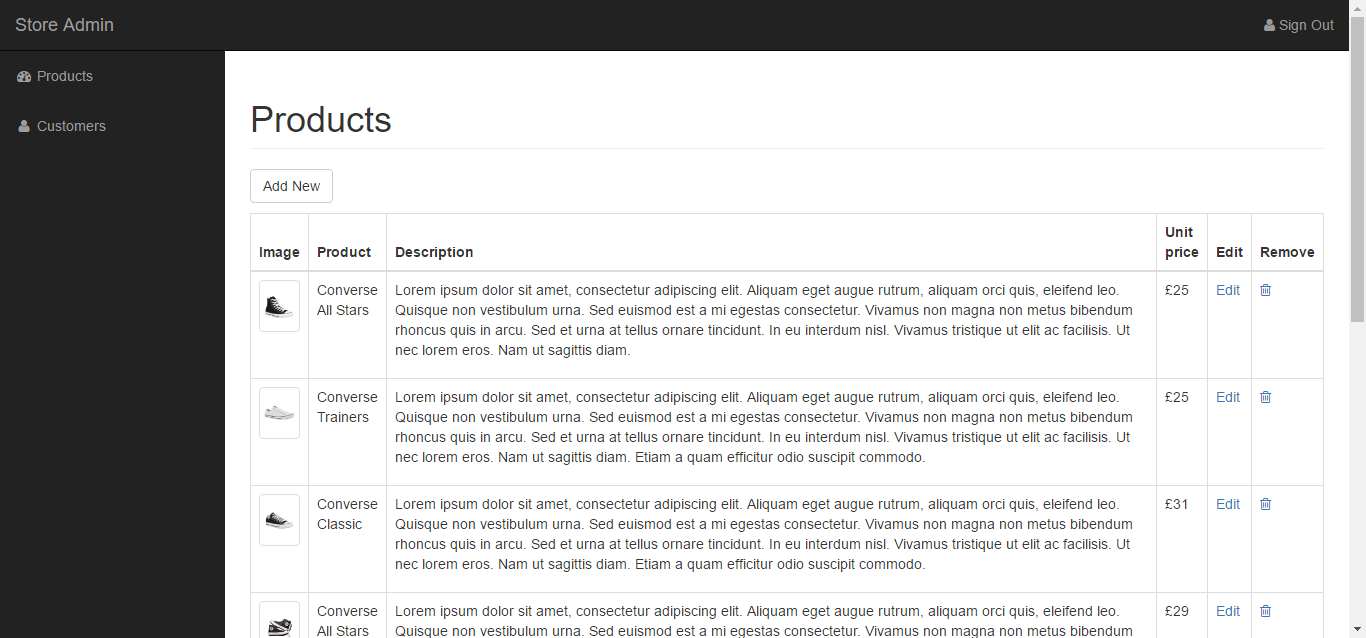
## Admin Panel

Location: http://localhost:5000/admin

The username is “admin” , password is “admin”

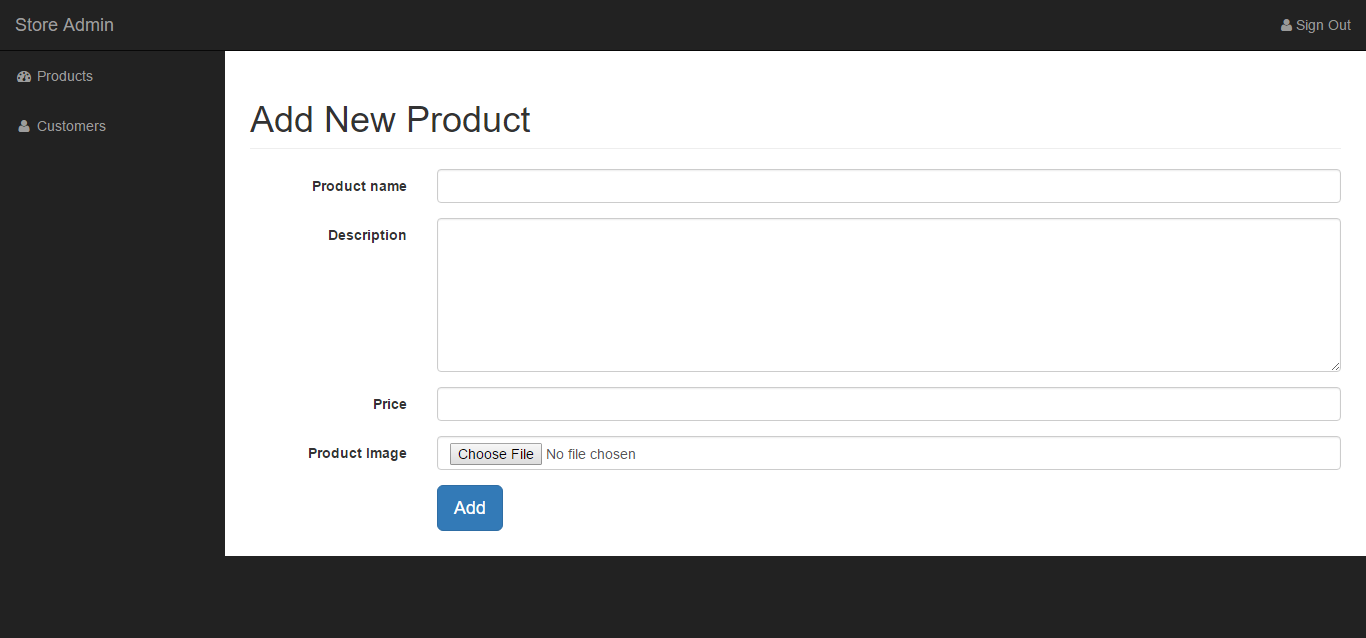


After logging in the user can browse, add and edit products or browse and delete users.



## Admin Add page

This is where the admin can upload new items to the database.



# Design and architecture

## Folder structure

I used SQLite to store information. The database file is “db/shop.db”

The database, logging, secret key, admin credentials and basic setup are configured in: “etc/config.cfg”

All CSS files such as bootstrap, owl carousel, font awesome and custom theme are in “static/css”

All JS files are in “static/js”

All images are in “static/img”

The log file can be found in “var/logging.log”

All HTML templates are in the “templates/” folder

The app can be started by running the index.py file

## URL structure

/route

/register

/cart

/product

/detail

/search

/admin

/admin\_edit

/admin\_add

/login

# Enhancement

## Database

* The database could be extended to more tables, such as size and stock; these tables could be joined together.
* The product table could store more images and related items for suggestion.
* To complete the e-commerce site we need order and payment table with the relevant information for check out.
* Add encryption to user password

## Front-end and functionalities

* Although I am using bootstrap as a responsive framework I could further optimise the mobile experience.
* Add recommended or suggested items on the detail page.
* On the home and the product page add item size to product details so it can be added straight away to the shopping cart.
* Add confirm password to register form and password validation with JS
* Add form validation for the Admin\_add page, File extension checker for images (png or jpg only)
* Add error messages or success messages on file upload
* Check image size on upload; also add id number to the files for better organisation.
* Add image change feature to Admin\_edit page so the user can change more than just a metadata
* Add pagination to details page or Admin page
* Deploy the app

## Shopping Cart

* At the moment if the user adds the same product with the same size it appears as separate items in the cart, this could be added to one product with increased quality.
* Add quantity update to the cart.
* Check out could be added with paypal test account.

# Critical evaluation

I believe I managed to cover most aspects what we learnt from the workbook. Overall I am satisfied with the final website as it has all the features that were required be the assessment but as I mentioned above there are many things that could be improved. I realised building an e-commerce website is a huge project. My website only covers the basic functionality of a web shop. Adding check out and Paypal gateway would be a good start along with a database extension. In a real word situation I would need to add a stock control system which would automatically update the stock on sale or return and wouldn’t display items or sizes that are not available. Security would play an important role as well so I have to make sure all customer and order details are stored securely as well as all payments are securely processed.

# Personal evaluation

I feel this project wasn’t very different from my first assessment. There were things that I wanted to improve in my previous one such as the file upload but unfortunately I faced some unexpected issues that took away my time to finish all the final touches.

The biggest challenge at this project was the user registration and the shopping cart. I am happy with the user registration as it has validation but the shopping cart could be further improved as I already mentioned above.

I feel I am way more comfortable to use Levinux and the command line as well as the Vim Editor.

I tried to use Jinja for some value calculation but at the end I only managed to do it with python or JavaScript so I feel I need to research this area more.

I definitely spent more time with the database design on this project. Originally I wanted to try some new database system, ideally a NoSQL database but speaking to my fellow classmates I decided not to go with it as there are numerous issues running them in the development environment as well as I found way less documentation regarding this topic than related to SQlite. I feel at the end SQLite did the job and I also got more comfortable using SQL databases.

I am happy that we are using GitHub for our coursework. We started to use it to our group project as well and it provides us a clear version controlling where we can track everyone’s work.

My code has improved since the previous project but still could be a bit more logical. I previously mentioned that I used a lot of inline CSS so this time I tried to avoid this.

# Resources

## Frameworks

**Bootstrap**

Type: Front end framework

URL: <http://getbootstrap.com/>

License: MIT

**OWL Carousel**

Type: Slider

URL: <http://owlgraphic.com/owlcarousel/>

License: MIT

**Font Awesome**

Type: Font/Special characters

URL: <http://fontawesome.io/>

License: MIT

**DB Browser for SQLite**

Type: Software

URL: <http://sqlitebrowser.org/>

License: GNU

**DB Browser for SQLite**

**Bootwatch**

Type: Stylesheet

URL: <https://bootswatch.com/>

License: MTI

**Fron End Framwork**

**Mimiti**

Type: HTML Template

URL: <https://wrapbootstrap.com/theme/mimity-online-shop-template-WB094DPGC>

License: Single

## References

Bootstrap documentation: [http://getbootstrap.com/](http://getbootstrap.com/%20)

Python Flask handbook: <https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/Python3_Flask.html>

<https://docs.python.org/2/library/sqlite3.html>

Jinja2 documentation: [http://flask.pocoo.org/](http://flask.pocoo.org/%20)

SQLite3[: http://flask.readthedocs.org/en/0.9/patterns/sqlite3/](file:///C:\Users\viragjacint\Dropbox\Uni\Webtech\coursework1\report\:%20http:\flask.readthedocs.org\en\0.9\patterns\sqlite3\)

Various examples: <http://stackoverflow.com/>