**MediaMarkt Service Quality**

**Review System**Top of Form

Media Markt is a German multinational chain of stores selling consumer electronics with over 1000 stores in Europe. This website is designed to get customer review on the store departments in Poland and Germany.

**The database scheme**

A picture containing text

Description automatically generated

Bottom of Form

**USER MANUAL**

The website’s interface is user friendly.

Graphical user interface, text, application

Description automatically generated

Every page has clear instructions. The visitor first sees the main page. The page includes instructions such:

* Please click on the **Submit a Review** button to review MediaMarkt Departments.
* [**Review submission**](http://localhost:8000/add/) and [**Review list**](http://localhost:8000/list) are only available for the **registered**users. You will be required to [**log in**](http://localhost:8000/login/) to your account.
* Unregistrated visitors may see [**Mediamarkt rating**](http://localhost:8000/average/) by **city**, **store** and **department** based on customer reviews. Navigation bar includes a button to refer official website, Login and Register.

A screenshot of a computer

Description automatically generated

* If a visitor wants to submit a review faces a login page. If visitors does not have an account, he/she can sign up. For that, there is a link to go the registration page.
* Graphical user interface, application

  Description automatically generatedThe registration has required fields: Username, E-mail, Password. Password has minimum requirements for security reasons.
* Graphical user interface, website

  Description automatically generatedAfter signing up, the user automatically is being sent to login page, where he/she can see a success message about signing up.
* After loggining in, User goes to the homepage automatically. User may know submit a review.

Graphical user interface

Description automatically generated

* Submit a review button takes a user to the Review Form page.
* Here a user should choose Country, City, Store and the Department.
* After that a user may rate the department in a scale of 5. The scale is based on the star rating (1 to 5 Star)
* Every field is required.
* Country, City, Store fields are dependent on each other.

Graphical user interface

Description automatically generated

* An example of filled form.
* Graphical user interface, application, Teams

  Description automatically generatedAfter review submission, user goes to the review list page. This page includes all the reviews by the users.
* In this page, user may see all the reviews by himself and other users.
* The reviews are sorted by date. The newest review are on the top.
* User may edit his own review but not the others. The page includes further instruction on editing/deleting.
* Graphical user interface

  Description automatically generatedAny visitor may visit Mediamarkt ratings page which includes ratings by city, store and department.
* The system automatically calculates the average ratings based on the submitted reviews.
  + For cities, stores and departments separately

Graphical user interface, website

Description automatically generated

* Profile page provides a user to change his username, email and profile photo.
* Graphical user interface, text, application

  Description automatically generatedUpon signing up, every user gets an default photo.
* The logout page shows a success message and also offers to log back in through a hyperlink.

**QUERIES**

Both – ORM and Raw queries has been used in the project. Below you may see the Raw queries has been used in the view.py file for calculate and group the average ratings by city, store and departments.

———————

**CITY Average**

select name, city\_id, avg(rating\_id) from blog\_review

join city

on blog\_review.city\_id = city.id

group by name, blog\_review.city\_id

——

**STORE Average**

select name, store\_id, avg(rating\_id) from blog\_review

join store

on blog\_review.store\_id = store.id

group by name, blog\_review.store\_id

—————————

**Department Average**

select name, department\_id, avg(rating\_id) from blog\_review

join departments

on blog\_review.department\_id = departments.id

group by name, blog\_review.department\_id

view.py file

def about(request):

connection = sqlite3.connect('Mediamarkt.db')

cursor = connection.cursor()

print("Connected to SQLite")

sqlite\_select\_query = """select name, avg(rating\_id) from blog\_review

join city

on blog\_review.city\_id = city.id

group by name, blog\_review.city\_id """

cursor.execute(sqlite\_select\_query)

records = cursor.fetchall()

connection.commit()

connection.close

##Store

connection = sqlite3.connect('Mediamarkt.db')

cursor = connection.cursor()

print("Connected to SQLite")

sqlite\_q = """SELECT name, avg(rating\_id) from blog\_review

join store

on blog\_review.store\_id = store.id

group by name, blog\_review.store\_id"""

cursor.execute(sqlite\_q)

records2 = cursor.fetchall()

connection.commit()

connection.close

##Department

connection = sqlite3.connect('Mediamarkt.db')

cursor = connection.cursor()

print("Connected to SQLite")

sqlite\_q = """select name, avg(rating\_id) from blog\_review

join departments

on blog\_review.department\_id = departments.id

group by name, blog\_review.department\_id"""

cursor.execute(sqlite\_q)

records3 = cursor.fetchall()

connection.commit()

connection.close

return render(request,'blog/average\_list.html',{'data2': records2, 'data': records, 'data3': records3})

This project is solely done by me – Ismayil Ismayilov 444459