COMPINTEL: A step for smart shopping

Rohit Nisal
MSc in Cloud Computing
National College of Ireland,
Dublin, Ireland
X16146271@student.ncirl.ie

Piyush Narkhede
MSc in Cloud Computing
National College of Ireland,
Dublin, Ireland
x17151538@student.ncirl.ie

***Abstract* – Shopping over internet is being now-a-days. People are preferring online website for shopping, which is better than went to store. So that e-commerce business is in high demand. This web application will help people to shop online. Our web application is connecting multiple shopping malls on single platform and user can choose their product from where they want to buy. This report is for showing some security facts regarding our application. This application also showing product siblings by help of recommendation system used in this application.**

Keywords- internet; e-commerce; shopping malls; recommendation system;

I. INTRODUCTION

COMPINTEL is the combination of “Compra Inteligente” which means smart shopping. We developed application which is based on online shopping. As we can see, people diverted towards online shopping rather than shopping in store. So, all websites like amazon.com, flipkart.com, ebay.com, etc. are getting popular for shopping. We planned to develop one application which contains multiple shopping malls on single platform.

Shopping malls are submitted their product to our admin and admin is going to add every shopping mall with their products on our application. Our application has some functionality which it showing which is cheapest option and some recommendations on basis of popularity.

We found that lots of users shop online but for same city, they are depending on local stores for regular use products. Even they randomly go to any shopping mall and purchase whatever they want. But same item is in cheap rate in different mall but how they know? They wasted more money in same item which they supposed to take from another mall. So, by this scenario we developed an application which will compare same product from different shopping malls and show cheapest rate product with their mall name. We are trying to show some recommendation on basis of previous ordered items. People can take benefit of our application like first check rates at their home and then go for shopping or they can book online so that they can shop online itself.

We are used Ruby and rails web framework which is famous for designing web applications. It is fully object-oriented programming language so that the web application will be secured. Objects are present for every transaction in Ruby. It uses sqlite3 engine for database operation.

It uses MVC architecture to develop application. Models are used to give relations for application. Views are using for webpage development and controller are using to control flow of application. If you want to debug, develop or change anything you can easily change by using this architecture. It provides testing facility so that you can easily test application by using their gems and some predefined coding.

II. SYSTEM ARCHITECTURE

Our application contains the entity relationship between the components in the application. The main featured entities are admin, user, product, order, profile, etc. are some of the basic tables used in the application. Each table is having some relationship to access their data like one to one, many to one, one to many and many to many which can be easily be demonstrated in the architecture diagram.

For the user registration, we are using devise gem which is having basic design structure for table i.e. it only stores the e-mail address and password, which is then associated with the profile table which gave the access to modify and add new content for user details which contain extra information like first name, last name, date of birth gender and address.

User must register first then create profile with their details. On home page they can access items present in our application. The can see which product is cheap and which is costlier with their availability. They can order from our application.

The product table stores the detailed information for any product, while adding some product the admin has the responsibility to fill the entire information for the product.

III. Specification of Project

A. *Create account:*

User has to register if they want to use our application. User must register with their email ID and password.

B. *Create Product:*

Admin had access to add products in our application. Admin added each product with their description and name with different mall names.

C. *Maintain Profile:*

After registration user have to create their profile with providing personal details in our application.

D. *Order Item:*

User can see different items in our application with order facility. Once they select any item just click on order, item will be ordered successfully.

IV. Project Implementation

A. *Application UI:*

For user interface we used two different types of styles. Piyush used CSS for creating sign in and sign up page with specific style and design. Rohit developed admin dashboard by using bootstrap. Another header and footer design is developed by using different bootstrap theme.

B. *Admin Profile:*

This functionality developed by Rohit. He used devise gem to secure admin login with authorization and authentication with admin rights. Also added one more functionality that admin can add products in our application. Admin can also see the order details which user requested for any product.

C. *Order Item:*

This was entirely carried out by Rohit. the product listed on front end/ web application can be ordered with certain quantity, which is then diverted to another page by clicking on a button in header which notify the number of item ordered and with the total price and the details are transfer to order table.

D. *Captcha Varificaton:*

Captcha verification is done by Rohit. Before giving order, captcha verification done by application. So, nobody can order items continuously. He created one separate gem for captcha which sending random mathematical sum with answer.

E. *User Profile:*

This function created by Piyush. Anybody can register in our application as user. They have to create profile which will be unique for application. User can maintain their profile by having edit option. No one can see another profile without permission. Security is provided for profiles even admin also never access user profile at all.

F. *Home Page:*

Home page is developed by Piyush. In home page, user can see different products having different rates from different malls. Even they can compare price between different malls and order whichever they want.

G. *Finding cheapest product:*

This function is developed by Piyush. He created a gem which takes all values of single product and by comparing those values, minimum value will be send on home page and it is easy to see cheapest product from another on home page. This gem is helping to finding cheapest rate of products over different malls. For large data, database has to fetch all these data and compare it for finding cheapest value, so easy solution he created a gem which reduces database transaction.

H. *Deploy Project:*

Deployment of project done by both Piyush and Rohit. We use Heroku for cloud application deployment. Heroku is one of the secure platform for deploying cloud application in the world.

I. Recommendation System:

Piyush is trying to develop this system. This system contains recommendations on basis of order given by users. When user orders any specific products together, order logs are generated. Based on this order table, this system will help user to purchase item sibling easily.

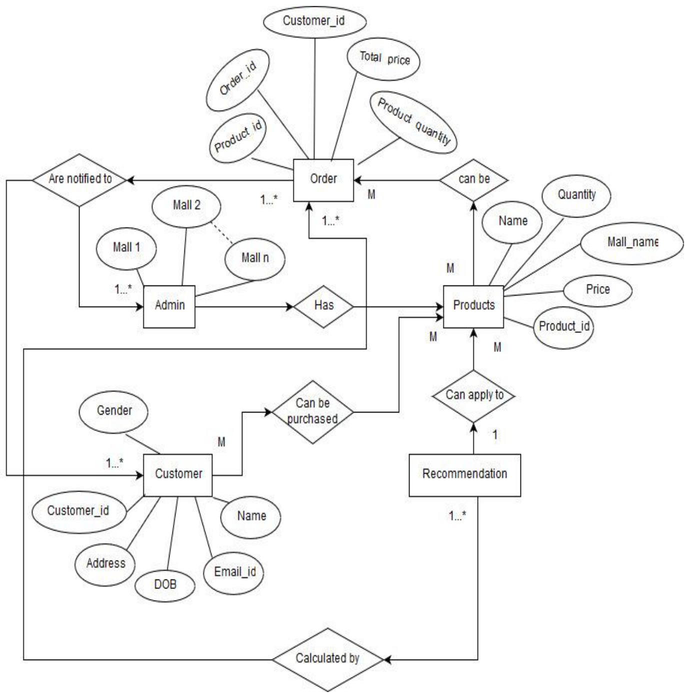


Fig.1: Entity Relationship Diagram for System Architecture

V. Library

A. Devise:

We are using devise for authorization, authentication and access control purpose. Devise is providing secured login and signup facility for our application. Devise gem is provided by ruby on rails framework. It is popular gem with some functionalities for encryption and secure login. It does allow the security feature to our application so that no other user can access someone else accounts and does anything wrong, so there is no need to add extra security or additional feature to devise. In devise there are authentication factor for user and admin which is set by the using Boolean values.

B. Captcha:

For captcha verification Rohit developed a gem which contains whole logic of captcha. It contains arithmetic calculation can see the simple arithmetic calculation and user has to answer that sum. The answer will be send to controller and after matching answer, validation will be done, and order can have done successfully.

C. Bootstrap:

Bootstrap-sb-admin and bootstrap sass are the gem which helps to develop user interface in our application. This gem is user by both Rohit and Piyush. This gem helping to apply styles to for our application. The bootstrap helps in designing the web-pages so that the user gets attracted and it gives the look-n- feel for it.

D. Rails-observers:

It is gem used by both Rohit and Piyush. It helps to deploy observer pattern in our application. It is gem provided by ruby and it helps to deploy observer in model and the further implementation we have to do differently.

E. Recommendation System:

For recommendation system, Piyush is using recommendation gem which contains algorithm for recommendation system.

If any user bought milk and bread together and another bought bread and eggs together and much more. So, this system predicts user which product will be better if you are going to buy that product.

Example:

A (milk, bread, cake)
B (bread, egg)
C (bread, milk, chocolate)
Prediction is: D (milk, bread) bought together most times.

Finally, recommendation system fetches this data and predict which sibling is better. For next time, whenever a user wants to buy milk it will predict to buy bread also or vice versa.

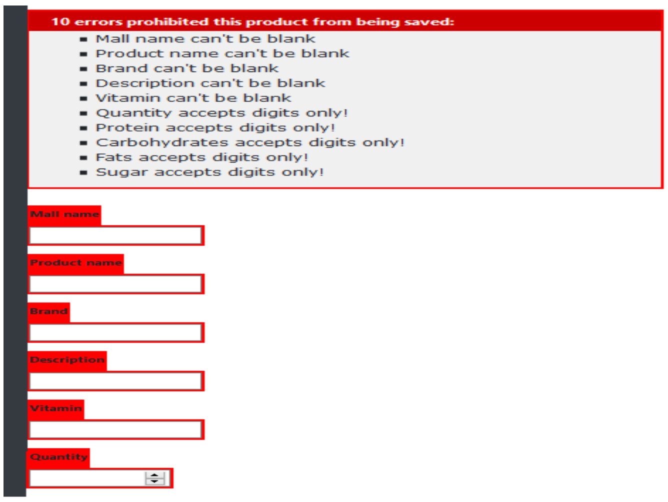
VI. Testing

For reliability of any application, testing is important factor which we must prepare in application. Testing help us to check all tags and functions whether all are in working or not. It helps to check whether application is working properly or there are any constraints which are making problems. It helps to debug our mistakes present in code. Some testing approaches for application testing:

A. *Unit Testing:*

Unit Testing contains testing of each text field in project. This validates whether provided inputs are correct.

Rohit tested his module by using unit testing. For adding product, admin must insert each text field correctly whether the input values are digit or alphabet depending on this we have inserted son regular expression which are used to valid the number and alphabets.



10 errors prohibited this product from being saved:

- Mall name can't be blank
- Product name can't be blank
- Brand can't be blank
- Description can't be blank
- Vitamin can't be blank
- Quantity accepts digits only!
- Protein accepts digits only!
- Carbohydrates accepts digits only!
- Fats accepts digits only!
- Sugar accepts digits only!

Mall name

Product name

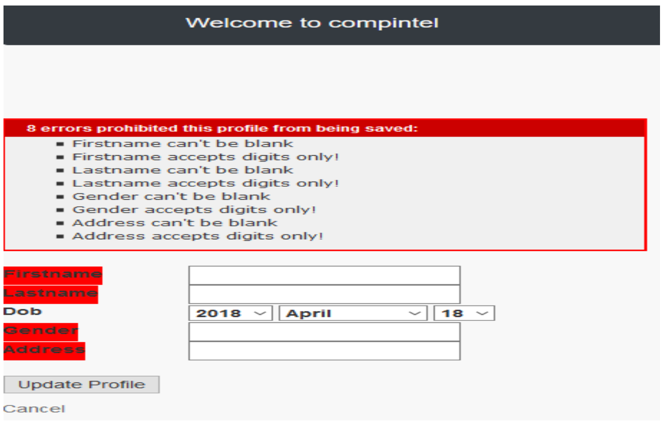
Brand

Description

Vitamin

Quantity

Piyush tested user profile create and update page with unit testing. Now no user can add wrong data in text field.



Welcome to compintel

8 errors prohibited this profile from being saved:

- Firstname can't be blank
- Firstname accepts digits only!
- Lastname can't be blank
- Lastname accepts digits only!
- Gender can't be blank
- Gender accepts digits only!
- Address can't be blank
- Address accepts digits only!

Firstname

Lastname

Dob 2018 April 18

Gender

Address

Update Profile

Cancel

B. *Integration Testing:*

Integration testing is doing after unit test but before system testing. This testing consists of different modules present in application. All modules taken into a group and tested in the group. So that it will result whether whole application is running properly or not.

C. *Functional Testing:*

Functional testing contains group of testes over single module. We can provide group of inputs at a module and check logs whether outputs are correct or not. We have to observer whether all functions are working properly or not. By giving some scripts

VII. Design Patterns

Design patterns helps to reduce loads present in software. We can reduce code and make use of object simple by using design patterns. We are also using some design patterns to implement some basic features and alert in our application. We are using observer and singleton patters in our application.

A. *Quantity Observer for Admin:*

Rohit developed observer pattern for giving notification to admin about quantity of product. If quantity is getting less, observer will send notification to admin that quantity is getting low please add more item.

B. *Singleton Pattern for Order Item:*

Rohit developed singleton pattern for creating order. User can order only one element at single time. It is easy to detect for admin side. Admin can easily get idea that for which mall, the order is placed.

C. *Quantity Observer for User:*

Piyush deployed observer pattern to getting know product is available or not. If quantity is 0 then observer showing message that product is no longer available.

D. *Singleton Patter for User Profile:*

Piyush used singleton pattern to maintain unique profile for each user. Every user has only one profile, he cannot able to create another separate profile for a single registration.

VIII. CHALLENGES

We deployed our application on Heroku cloud application platform successfully. We cannot able to import database from different malls so that this application is only for limited number of items. Because admin alone is going to add each item manually. In our application, there is system that user can order only single item at a single time so that we cannot predict any recommendation by that order table. So Piyush tries that recommendation system with manual order table created by him. And showed that as recommendation. This recommendation gem wants lots of data to predicts and our application is for limited use, so this application cannot ready for deploy in real market.

IX. CONCLUSION AND FINDINGS

As we can see, from customer view we tried to deploy one application for online shopping. People can use this application for their regular use. This application may save their money and so that they can shop more in same price. By the help of recommendation, people can easily order cheapest product from wherever they want to use.
By using MVC framework it gets easy to develop web application for us. Ruby is fully object-oriented language so that all transactions and methods are done by using object itself. If further we want to do change in this application, then we can easily change it because we are having MVC module with us.
We found that after deploying this application, we need to improve more complexities in this project so than we can add more functions and features like discount, online pay and many more.

REFERENCES

[1] Rohit(x16146271): Heroku Link [Online] Available at: https://murmuring-bayou-13126.herokuapp.com/

[2] Piyush(x17151538): Heroku Link [Online] Available at: https://protected-ocean-87416.herokuapp.com/

[3] Piyush(x17151538)Gem for comparison [Online] Available at: https://rubygems.org/gems/cmp_min_no/versions/0.0.0

[4] Rohit(x16146271): Gem for captcha [Online] Available at: https://rubygems.org/gems/captcha_img

[5] Start Bootstrap: SB-admin Bootstrap theme [Online] Available at: https://startbootstrap.com/template-overviews/sb-admin/ .

[6] Start Bootstrap: Scrolling Nav [Online] Available at: https://startbootstrap.com/template-overviews/scrolling-nav/ .

[7] GitHub: Implementation for devise [Online] Available at: https://github.com/plataformatec/devise .

[8] GitHub: Implementation for Recommendation [Online] Available at: https://github.com/id774/recommendation .

[9] Software Testing Class: Functional testing vs Non-functional testing [Online] Available at: http://www.softwaretestingclass.com/functional-testing-vs-non-functional-testing/ .

[10] GitHub: shopping_cart reference [Online] Available at: https://github.com/nawajlimon/shopping-carts

[11] GitHub: Implementation of Design pattern [Online] Available at: https://github.com/nslocum/design-patterns-in-ruby .