CHAPTER 1

1. INTRODUCTION

1.1 Introduction :-

The last decade has witnessed an explosion of data traffic over the communication network attributed to the rapidly growing cloud computing and pervasive mobile devices. This trend is expected to continue for the foreseeable future with a whole new generation of applications including 4K/8K UHD video, hologram, interactive mobile gaming, tactile Internet, virtual/augmented reality (VR/AR), mission-critical communication, smart homes, and a variety of IoT applications [1]. As the cloud infrastructure and number of devices will continue to expand at an accelerated rate, a tremendous burden will be put on the network. Thus, it is imperative for network operators to develop innovative solutions to meet the soaring traffic demand and accommodate diverse requirements of various services and use cases in the next generation communication network. Thanks to the economy of scale and supercomputing capability advantages, cloud computing will likely continue to play a prominent role in the future computing landscape. However, cloud data centers (DC) are often geographically distant from the end-user, which induces enormous network traffic, along with significant communication delay and jitter. Hence, despite the immense power and potential, cloud computing alone is facing growing limitations in satisfying the stringent requirements in terms of latency, reliability, security, mobility, and localization of many new systems and applications (e.g., embedded artificial intelligence, manufacture automation, 5G wireless systems) [1]. To this end, edge computing (EC) [2], also known as fog computing (FC) [1], has emerged as a new computing paradigm that complements the cloud to enable the implementation of innovative services right at the network edge. EC forms a virtualized platform that distributes computing, storage, control, and networking services closer to end-users to smarten the edge network. The size of an EN is flexible ranging from smartphones, PCs, smart access points (AP), base stations (BS) to edge clouds [3]. For example, a smartphone is the edge between wearable devices and the cloud, a home gateway is the edge between smart appliances and the cloud, a cloudlet, a telecom central office, a micro DC is the edge between mobile devices and cloud core network. Indeed, the distributed EC infrastructure compasses any computing, storage, and networking nodes along the path between end devices and cloud DCs, not just exclusively nodes located at the customer edge [3]. By providing elastic resources and intelligence at the edge of the network, EC offers many remarkable capabilities, including local data processing and analytics, distribute d caching, location awareness, resource pooling and scaling, enhanced privacy and security, and reliable connectivity

1.2 Existing System

Different from the existing literature, which mostly deals with optimizing the overall system performance from a single network operator's point of view, we consider the EC resource allocation problem from the game theory and market design perspectives. In particular, we study how to allocate resources from multiple ENs to multiple services in a fair and efficient way. We exploit the General Equilibrium, a Nobel prize-winning theory, to construct an efficient market-based resource allocation framework. Although this concept was proposed more than 100 years ago, only until 1954, the existence of an ME was proved under mild conditions in the seminal work of Arrow and Debreu. However, their proof based on fixed-point theorem is no constructive and does not give an algorithm to compute equilibrium. Recently, theoretical computer scientists have expressed great interests in understanding algorithmic aspects of the General Equilibrium concept. Various efficient algorithms and complexity analysis for ME computation have been accomplished over the past decade Note that although the existence result has been established, there is no general technique for computing an ME.

1.3 Proposed System

Our proposed models are inspired by the Fisher market which is a special case of the exchange market model in the General Equilibrium theory. An exchange market model consists of a set of economic agents trading different types of divisible goods. Each agent has an initial endowment of goods and a utility function representing her preferences for the different bundles of goods. Given the goods' prices, every agent sells the initial endowment, and then uses the revenue to buy the best bundle of goods they can afford. The goal of the market is to find the equilibrium prices and allocations that maximize every agent's utility respecting the budget constraint, and the market clears. In the Fisher market model, every agent comes to the market with an initial endowment of money only and wants to buy goods available in the market. We cast the EC resource allocation problem as a Fisher market. We not only show appealing fairness properties of the equilibrium allocation, but also introduce efficient distributed algorithms to find an ME. More importantly, we systematically devise a new and simple convex program to capture the market in which money has intrinsic value to the buyers, which is beyond the scope of the classical Fisher market model. Different from the existing works on cloud economics and resource allocation in general, our design objective is to find a fair and efficient way to allocate resources from multiple nodes (e.g., ENs) to budget-constrained agents (i.e., services), which makes every agent happy with her resource allotment and ensures high edge resource utilization. The proposed model also captures practical aspects, for example, a service request can be served at different ENs and service demands can be defined flexibly rather than fixed bundles as in auction models.

1.4 Advantages of proposed system

The greatest advantages of cloud computing are that it can allocate resources, maximize the use of cloud resources and minimize operating costs [4]. Cloud computing may continue to play an important role in future computing because of its economies of scale and supercomputing capabilities [5]. With the development of an open market for cloud computing resources, customers can easily obtain cloud computing resources from multiple cloud providers, which renders the interaction between multiple cloud providers and multiple customers a challenging problem

CHAPTER 2

2. REQUIREMENT ANALYSIS

2.1 Requirement Analysis

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

2.2 Software Requirement Specification

2.2.1 Functional requirement

- Graphical User interface with the User.

2.3 Resource requirements

2.3.1 Hardware requirements :-

✓ Processor	:	Pentium IV or higher
✓ RAM	:	2 GB
✓ Hard Disk	:	40 GB
✓ Mouse	:	Optical Mouse.
✓ Monitor	:	15’ Colour Monitor.

2.3.2 Software requirements :-

✓ Operating System	:	Windows 7 or Above
✓ Compiler / Interpreter	:	Python 3.7
✓ Libraries	:	Django
✓ Database	:	MySql
✓ Hosting Server Application	:	WampServer 2.4
✓ Brower	:	Chrome or Internet Explorer

CHAPTER 3

3. DESIGN

3.1 Design

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

3.2 Architecture

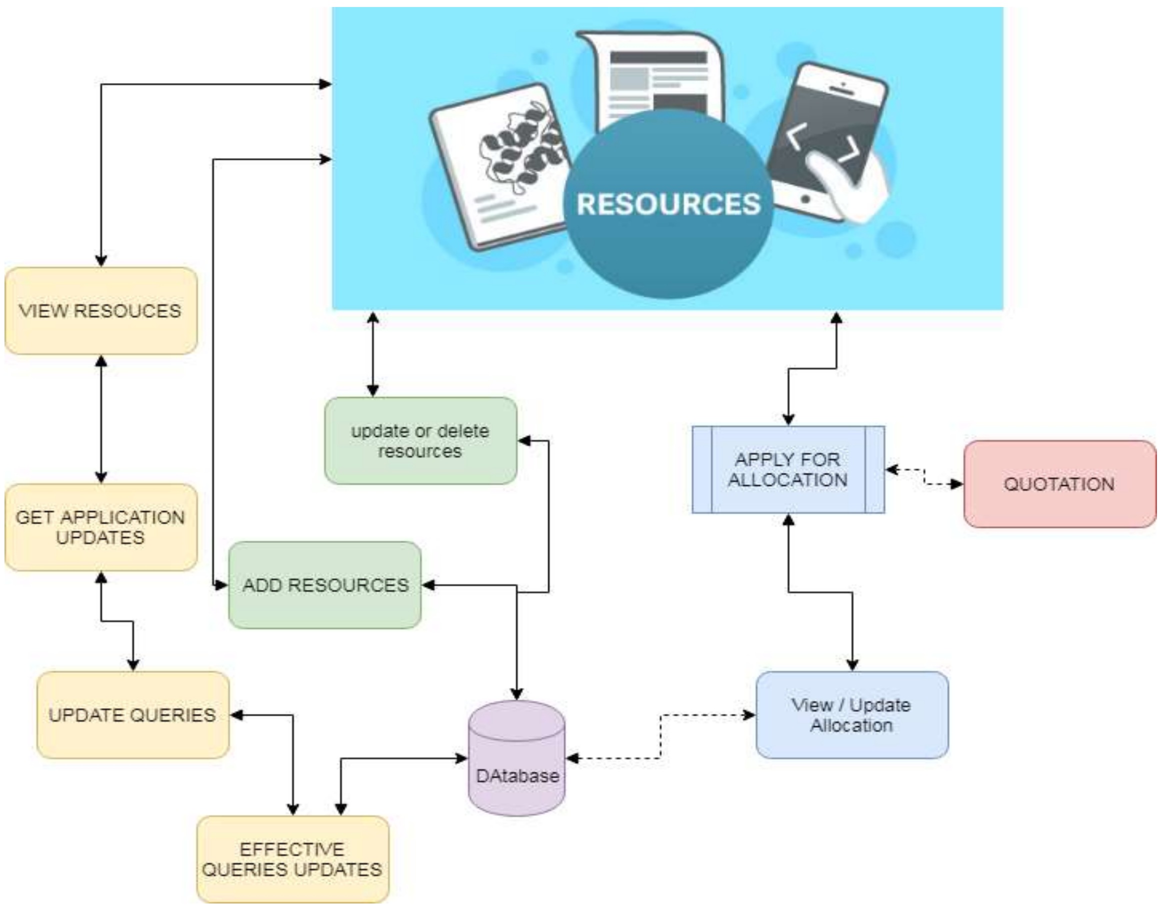


Figure 3.1 Architecture

3.2 Use case Diagram

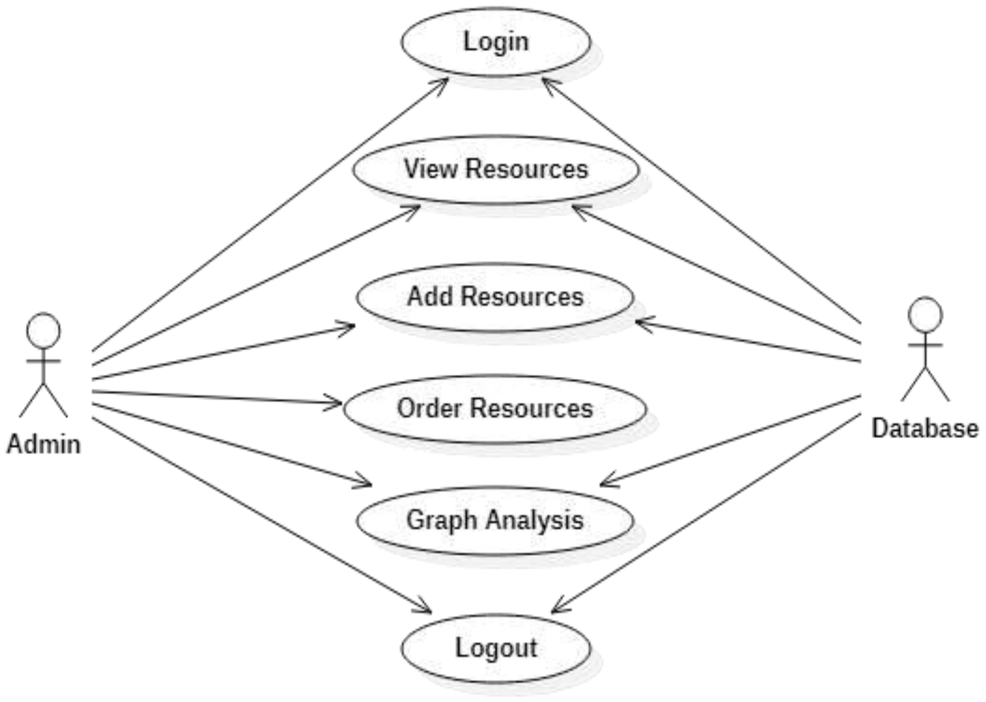


Figure 3.2 Use case Diagram

The use case diagram is used to represent all the functional use cases

3.3 Class Diagram

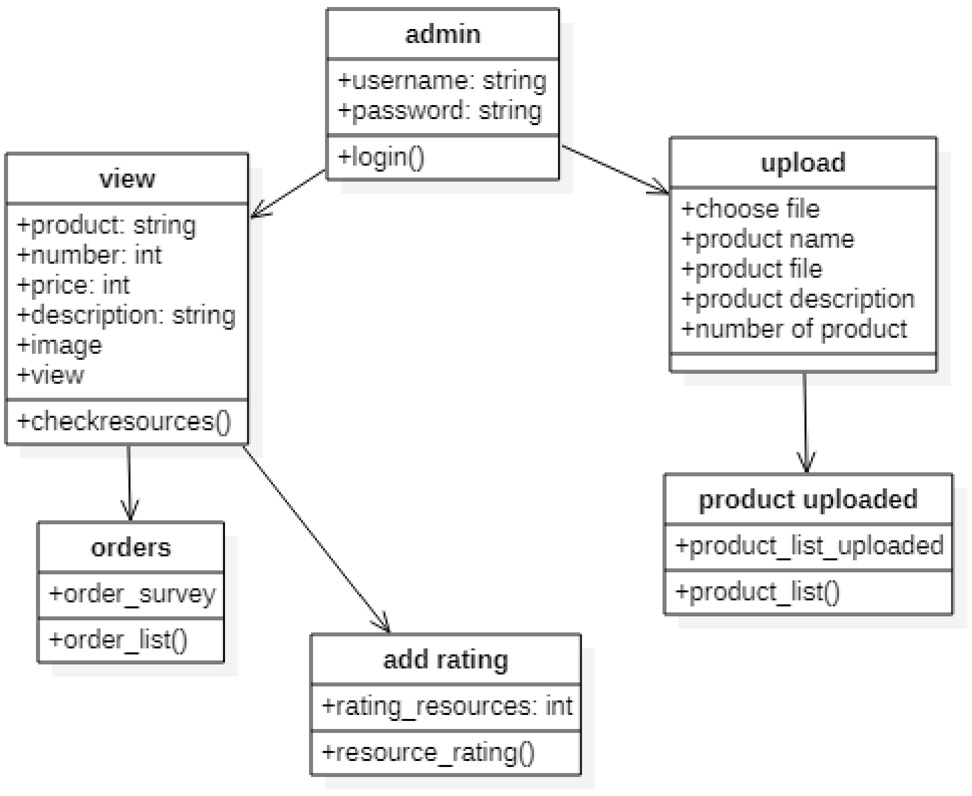


Figure 3.2 class diagram

The above mentioned class diagram represents the system workflow model.

3.4 Sequence Diagram

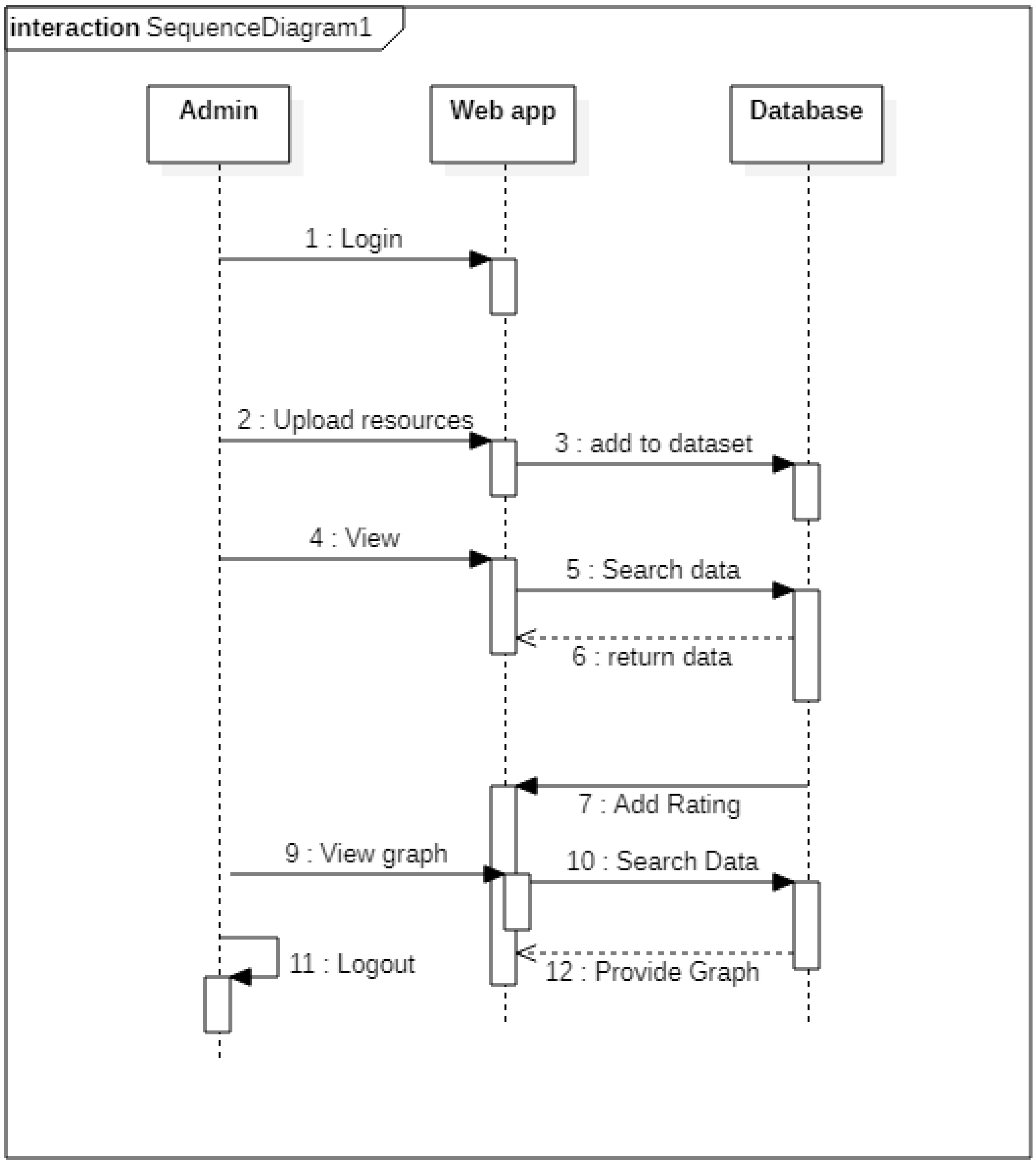


Figure 3.3 Sequence diagram

The below Sequence diagram represents the sequence of flow of actions in the system.

3.5 State Chart Diagram

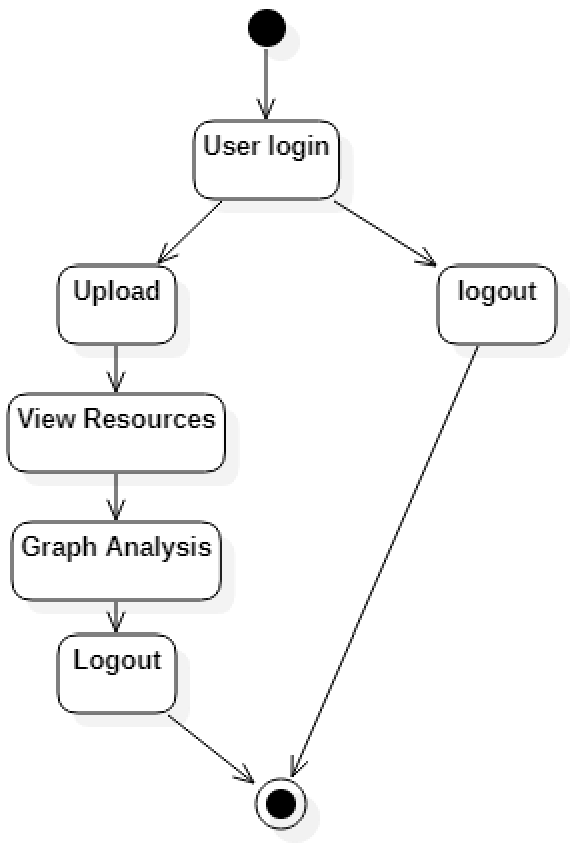


Figure 3.4 State Chart diagram

The below State Chart diagram represents the flow of actions in the system

CHAPTER 4

4. MODULES

4.1 Modules

1. ADD RESOURCE

   The resources have been uploading in database in order to view for users. Resources can be uploading, modify or delete. The added resources can be visible to user and user can apply with required details. User can add details with their quotation of application. So that user can implement their work.

2. ALLOCATE RESOURCE

   The received applications are viewed by admin. Admin then, find the available space and view application and based on that, algorithm applied to sort the best user to be allocated or the allocation space will be measured according to the quotation of user had submitted. The allocated resource can be utilized by user and admin need to intimate the user that how much resource have been allocated.

3. USER QUERIES

   Users can have queries about the process. This part of project is dedicated to make and get response for queries that are needed to answerable. The major part of the modules is making project as interactive one, queries have been very normally arise to users regarding different details about the process.

4. GRAPH ANALYSIS

Graph analysis is the part where admin can knows the statistics about process of details. The data are taken from the project flow and it shows until updated value. The data are gives clear solution to admin that part of improvement and user satisfaction and other factors.

CHAPTER 5

5. IMPLEMENTATION

5.1 Sample Code

```
#!/usr/bin/env python
import os
import sys
if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"findingtrust.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

5.2 Database Tables

5.2.1 Auth Group

Table No 5.1

```
CREATE TABLE IF NOT EXISTS `auth_group` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(80) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;
```

5.2.2 Auth Permission

Table No 5.2

```
CREATE TABLE IF NOT EXISTS `auth_permission` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `content_type_id` int(11) NOT NULL,
  `codename` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `auth_permission_content_type_id_codename_01ab375a_uniq`
(`content_type_id`,`codename`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=31 ;
```

5.2.3 Consumer Order

Table No 5.3

```
CREATE TABLE IF NOT EXISTS `consumer_orders` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `prodqty` int(11) NOT NULL,
  `location` varchar(200) NOT NULL,
  `user_id` int(11) NOT NULL,
  `prod_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `consumer_orders_user_id_90bd03f8_fk_general_profile_id` (`user_id`),
  KEY `consumer_orders_prod_id_6b4563f7_fk_serviceprovider_product_id`
(`prod_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```

CHAPTER 6

6. SCREENSHOTS

6.1 Screenshots.

6.1.1 Login Page

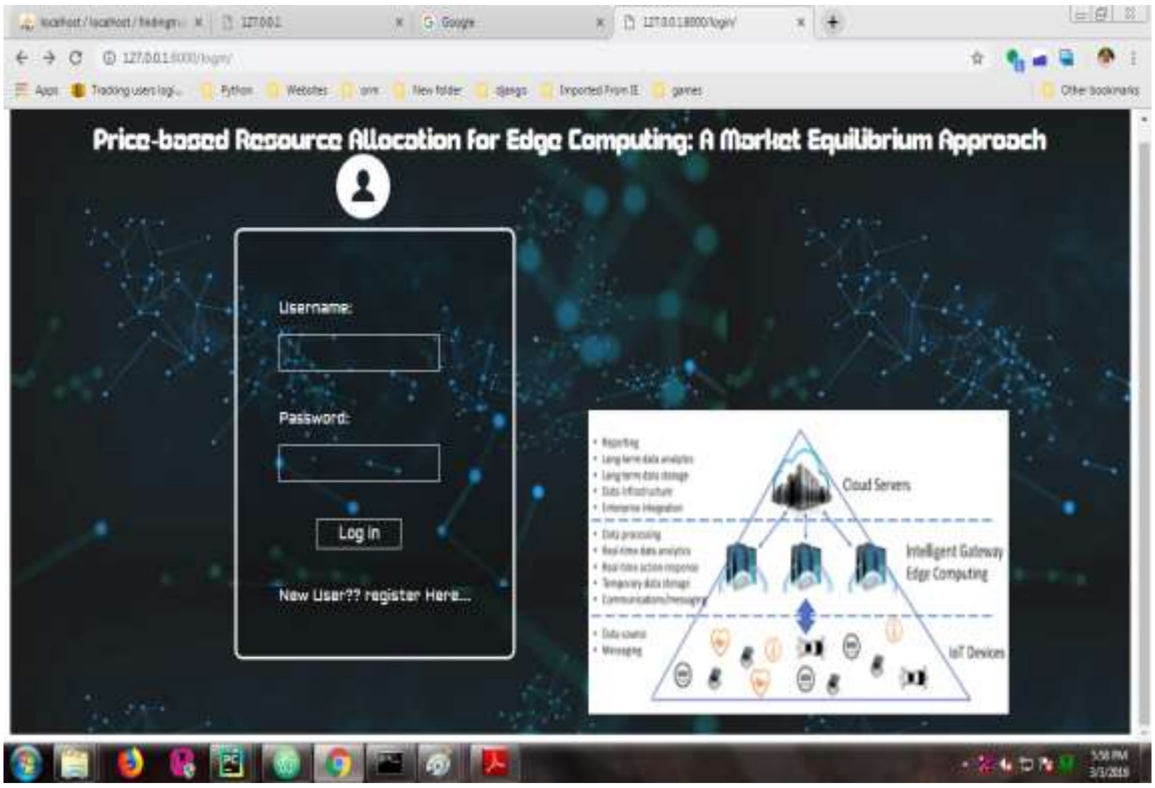

Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach
Username:
Password:
Log in
New User?? register Here...
Cloud Servers
Intelligent Gateway Edge Computing
IoT Devices

Figure No. 6.1 Login Page

Screenshot show the Login page

6.1.2 Registration Page

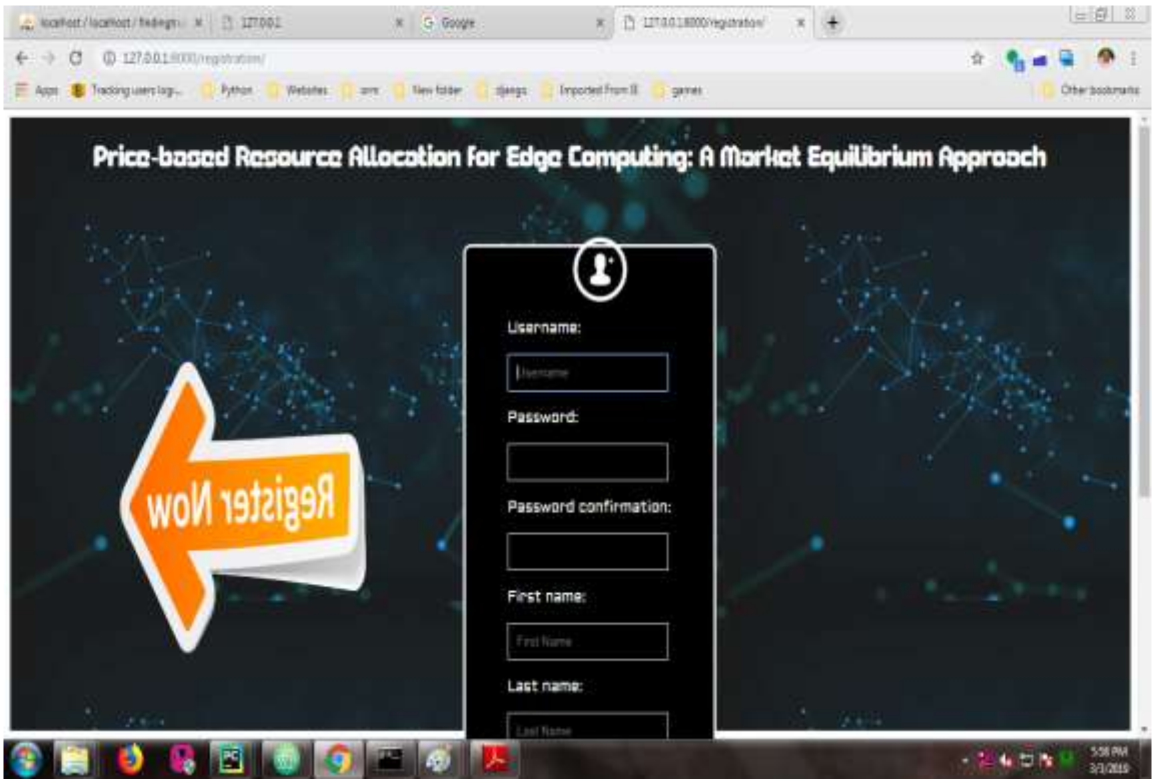
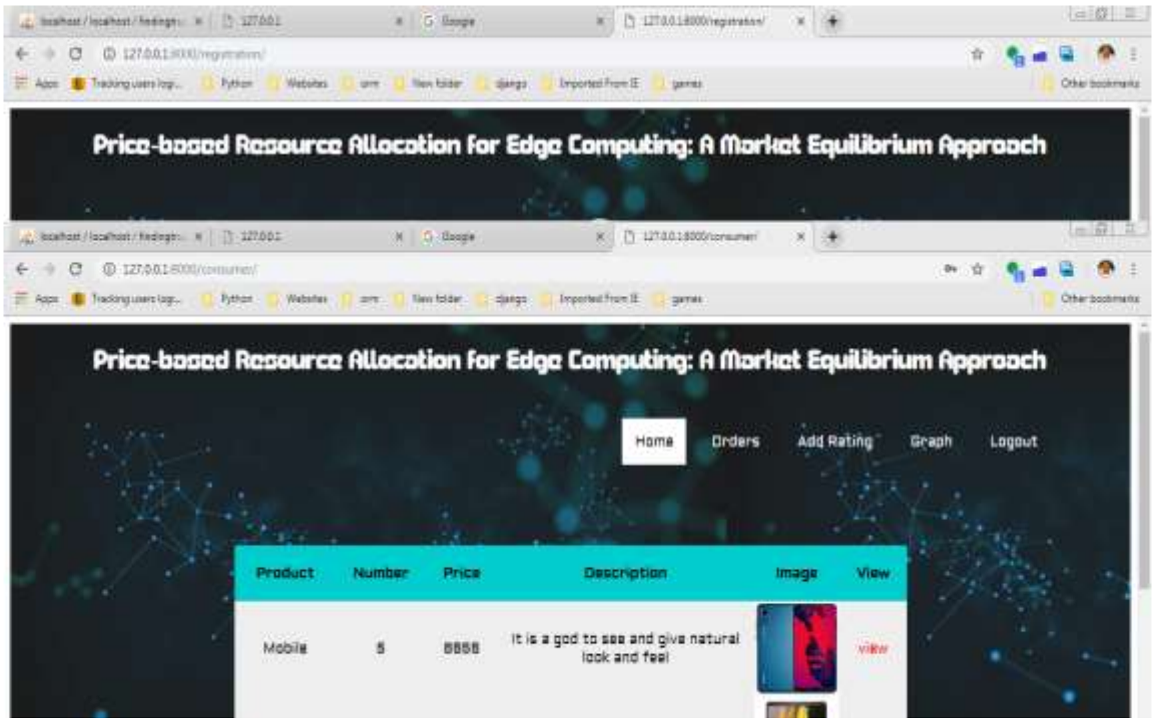

Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach
Register Now
Username:
Password:
Password confirmation:
First name:
Last name:

Figure No. 6.2 Registration Page

Screenshot show the Registration page

6.1.3 Home Page



Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach

Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach

Product	Number	Price	Description	Image	View
Mobile	5	8888	It is a god to see and give natural look and feel		view

Figure No. 6.3 Home Page

Screenshot show the Home page

6.1.4 Add Product Page

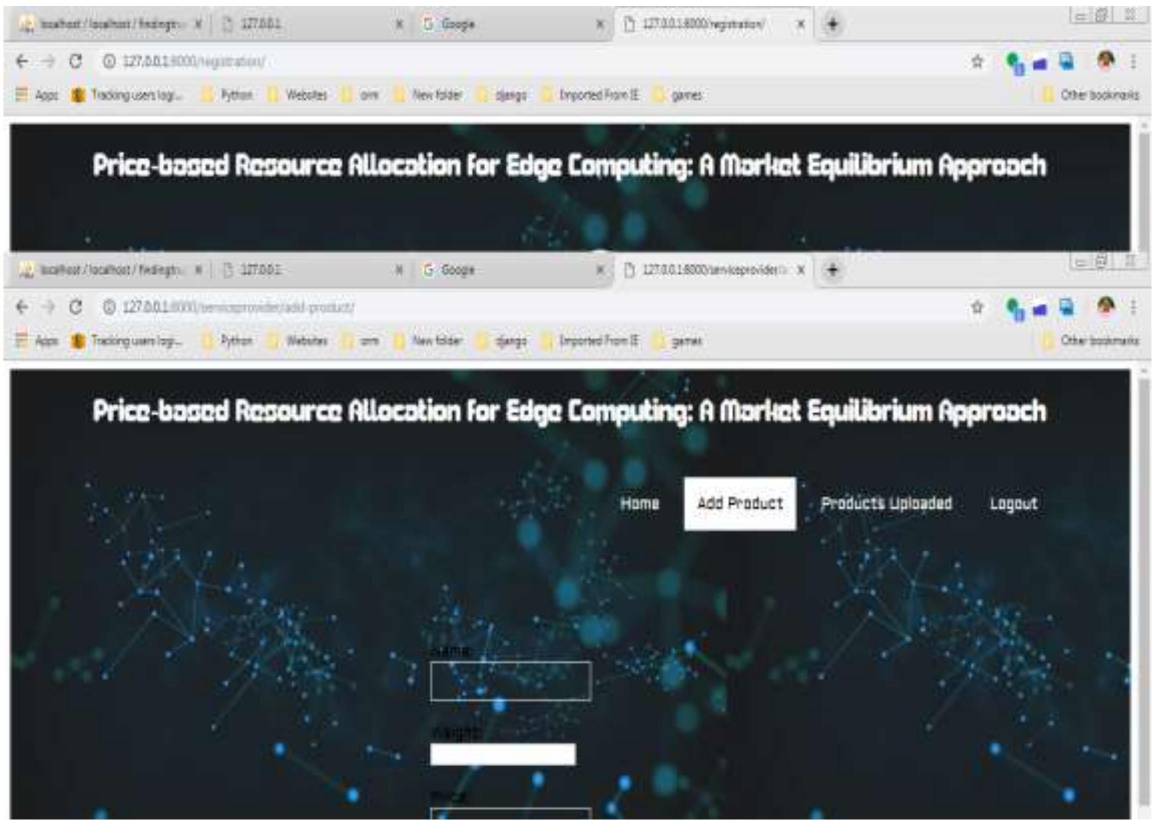

Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach
Home
Add Product
Products Uploaded
Logout
Name
Weight

Figure No. 6.4 Add Product Page

Screenshot show the add product page

CHAPTER 7

7. TESTING

7.1 Overview of Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 Types of Testing

7.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.3 Unit Testing :-

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.4 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.5 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 8

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

In this work, we consider the resource allocation for an EC system which consists geographically distributed heterogeneous ENs with different configurations and a collection of services with different desires and buying power. Our main contribution is to suggest the famous concept of General Equilibrium in Economics as an effective solution for the underlying EC resource allocation problem. The proposed solution produces an ME that not only Pareto-efficient but also possesses many attractive fairness properties. The potential of this approach are well beyond EC applications. For example, it can be used to share storage space in edge caches to different service providers.

We can also utilize the proposed framework to share resources (e.g., communication, wireless channels) to different users or groups of users (instead of services and service providers). Furthermore, the proposed model can extend to the multi-resource scenario where each buyer needs a combination of different resource types (e.g., storage, bandwidth, and compute) to run its service. We will formally report these cases (e.g., network slicing, NFV chaining applications) in our future work.

8.2 Scope for future work

The proposed framework could serve as a first step to understand new business models and unlock the enormous potential of the future EC ecosystem. There are several future research directions. For example, we will investigate the ME concept in the case when several edge networks cooperate with each other to form an edge/fog federation. Investigating the impacts of the strategic behavior on the efficiency of the ME is another interesting topic. Note that N. Chen et. al. have shown that the gains of buyers for strategic behavior in Fisher markets are small. Additionally, in this work, we implicitly assume the demand of every service is unlimited. It can be verified that we can add the maximum number of requests constraints to the EG program to capture the limited demand case, and the solution of this modified problem is indeed an ME. However, although the optimal utilities of the services in this case are unique, there can have infinite number of equilibrium prices. We are investigating this problem in our ongoing work. Also, integrating the operation cost of ENs into the proposed ME framework is a subject of our future work. Finally, how to compute market equilibria with more complex utility functions that capture practical aspects such as task moving expenses among ENs and data privacy is an interesting future research direction

REFERENCES

[1] M. Chiang and T. Zhang, “Fog and IoT: an overview of research opportunities,” IEEE Internet Things J., vol. 3, no. 6, pp. 854–864, Dec. 2016.

[2] M. Satyanarayanan, “The emergence of edge computing,” Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: vision and challenges,” IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] K.J. Arrow and G. Debreu, “Existence of equilibrium for a competitive economy,” Econometrica, vol. 22, no. 3, pp. 265–290, 1954.

[5] W.C. Brainard and H.E. Scarf, “How to compute equilibrium prices in 1891,” Cowles Foundation, Discussion Paper, no. 1272, 2000.

[6] A. Mas-Colell, M. D. Whinston, and J. R. Green, “Microeconomic Theory”, 1st ed. New York: Oxford Univ. Press, 1995.

[7] H. Moulin, “Fair division and collective welfare,” MIT Press, 2004.

[8] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, “Algorithmic Game Theory”, Cambridge, U.K.: Cambridge Univ. Press, 2007.

[9] E. Eisenberg and D. Gale, “Consensus of subjective probabilities: The pari-mutual method,” Annals of Mathematical Statistics, vol. 30, pp. 165–168, 1959.

[10] E. Eisenberg, “Aggregation of utility functions,” Manage. Sci. 7, PP. 337–350, 1961.

[11] Y. Lin and H. Shen, “CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service,” IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 2, pp. 431–445, Feb. 2017