IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

VALUE-BASED RESOURCE ALLOCATION FOR EDGE COMPUTING: A MARKET BALANCING APPROACH

MANYAM THAILE[1], PRAMATH PARASHAR[2], J HEMANTH[3], RITHIKA SOMANNGARI[4]

[1-4]MALLA REDDY ENGINEERING COLLEGE (AUTONOMOUS), MAISAMMAGUDA(H), GUNDLAPOCHAMPALLY VILLAGE, MEDCHAL-MALKAJGIRI (DT), TELANGANA (S), INDIA

Abstract:

The outgoing computer paradigm promises to deliver advanced user information and enable a wide range of Internet of Things (IoT) applications. In this paper, we propose a new market-based framework for effectively allocating heterogeneous capacity-limited edge nodes (EN) resources to multiple competing resources at the end of the network. By setting the appropriate rates for locally distributed ENs, the proposed framework produces a market equity (ME) solution that not only increases computer utilization but also allocates a range of good resources to services because of budget constraints. If the use of the service is defined as the maximum revenue service can receive from its service allocation, equity can be computerized in one place by resolving the Eisenberg-Gale (EG) convex system. We also demonstrate that equity is Pareto-optimal and that it satisfies the demands of equity, including the sharing of motivation, equality, and non-envy. Also, two distributed algorithms, which successfully combine with ME, are developed.

Keywords:

Resource management, Cloud computing, Computational modelling, Edge computing, Delays, Reliability, Security

1. Introduction:

The last decade has seen the explosion of data traffic on the communication network caused by the rapid growth of cloud computing and mobile devices all over the place. The trend is expected to continue in the foreseeable future with a new generation of applications that include 4K / 8K UHD video, hologram, interactive mobile games, virtual internet, virtual/augmented reality (VR / AR), key goal communications, smart homes, and a variety of IoT applications.

2. System Analysis:

2.1 EXISTING SYSTEM:

Unlike the existing literature, which focuses on improving the overall system performance from a single network user perspective, we consider the EC resource allocation problem from game theory and market design ideas. In particular, we learn how to allocate resources from multiple ENs to

multiple services efficiently and effectively. We are using General Equilibrium, a Nobel Prize-winning theory, to create an effective framework for the distribution of market-based resources.

2.2 PROPOSED SYSTEM:

Our proposed models are inspired by the Fisher market which is a unique model of the General Equilibrium exchange market theory. The exchange market model consists of a set of economic agents trading in different types of separate assets. The goal of the market is to find the same prices and stocks that increase the resources of each agent concerning the budget limit, and the market is clear.

2.3 STUDY OF THE SYSTEM :

To provide flexibility for users, improved browser-enabled connectivity. Advanced GUIs are categorized as 1. User management interaction 2. Active or regular user interaction.

2.4 INPUT & OUTPUT REPRESENTATION :

Input design is part of the overall system design.

2.4.1 INPUT DESIGN :

Input design is the link between the information system and the user. The input design takes into account the following:

• What data should be provided as input?

• How should the information be organized or coded?

• Dialogue to direct staff active in providing input.

• Procedures for preparing for confirmation of installation and the steps to follow in case an error occurs.

2.4.2 OUTPUT DESIGN :

Quality output alone meets the needs of the end-user and presents the information. In any system, the results of the process are transmitted to users and another system automatically. It is a very important and direct source of information for the user. An effective and intelligent output design enhances system relationships to help make user decisions.

2.5 PROCESS MODEL USED WITH JUSTIFICATION :

2.5.1 SDLC (Spiral Model):

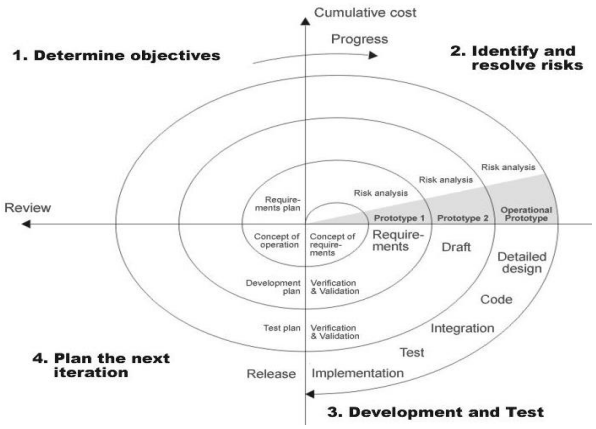


Figure-1 SDLC (Spiral Model)

SDLC is nothing but a Software Development Life Cycle. It is a standard used by the software industry to develop good software.

2.5.2 Requirements Gathering stage:

The collection process takes into account the inclusion of objectives identified in the project needs category of the project plan.

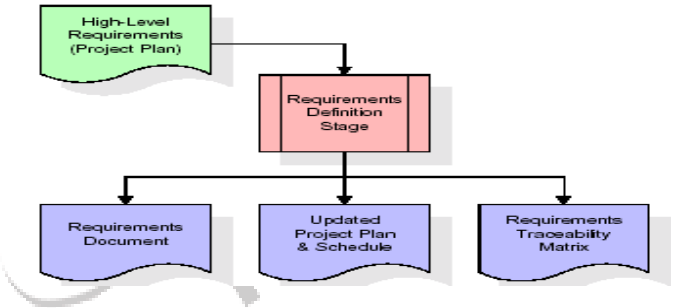


Figure-2 Requirements Gathering stage

These requirements are detailed in the main posts of this section: Requirements Document and Requirements Traceability Matrix (RTM).

2.5.3 Analysis Stage:

The planning phase establishes a bird's eye view of the intended software product and uses this to identify the basic project structure, assess the feasibility and risks associated with the work, and define appropriate management and technical procedures.

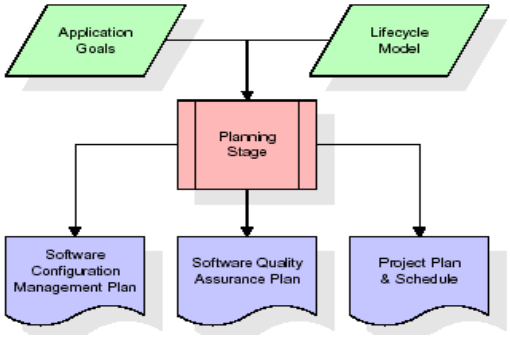


Figure-3 Planning Stage

The most important part of a project plan is a list of high-quality product requirements, also called objectives.

2.5.4 Designing Stage:

The design phase takes it as its first inclusion of the identified needs in the authorized requirements document. For each need, a set of one or more elements will be produced as a result of discussions, workshops, and/or exemplary efforts.

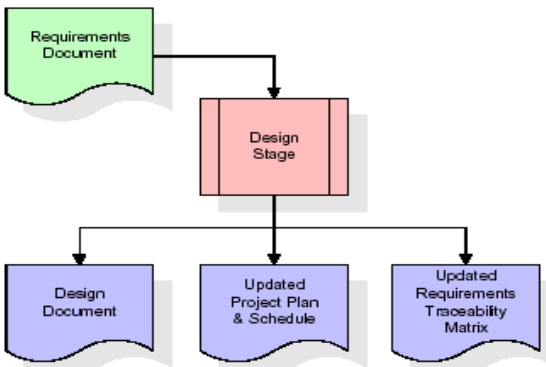


Figure-4 Design Stage

The results of the design phase of the design text, the updated RTM, and the new project plan.

2.5.5 Development (Coding) Stage:

The development phase treats as its main integration of the design elements described in the approved design document. For each phase of the project, one or more software sets of artefacts will be produced. Software artefacts include but are not limited to menus, interviews, data management forms, data reporting formats, and special processes and functions.

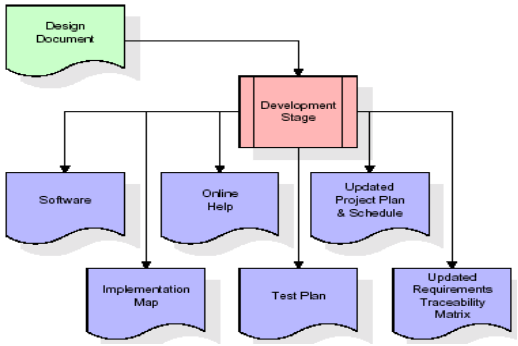


Figure-5 Development Stage

The RTM will be reviewed to indicate that each enhanced artefact is linked to a specific design object and that each enhanced artefact has one or more corresponding charges.

2.5.6 Integration & Test Stage:

During integration with the testing phase, software artefacts, online help, and test data are moved from the development site to a separate test site.

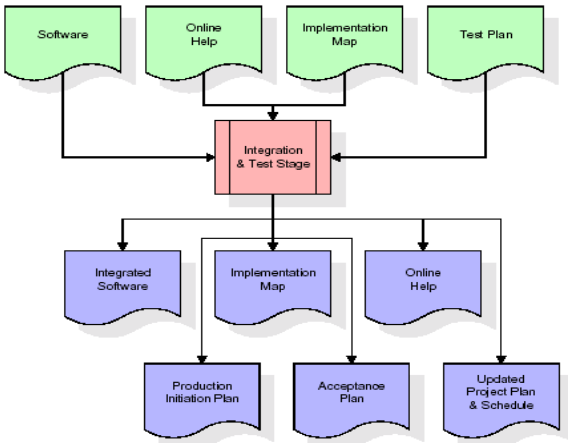


Figure-6 Integration & Test Stage

The results of the compilation and testing phase include an integrated software setup, an online help system, a startup map, a production startup program that describes reference data and production users, an acceptance program containing a final list of test cases, and a new project plan.

2.6 SYSTEM ARCHITECTURE:

2.6.1 Architecture flow:

The layout diagram mainly represents the flow of requests from users to the server via the server. In this case, the whole system is designed with three tires separately using three layers called the presentation layer, the business logic layer and the data connection layer. The project was developed using 3-tire architecture.

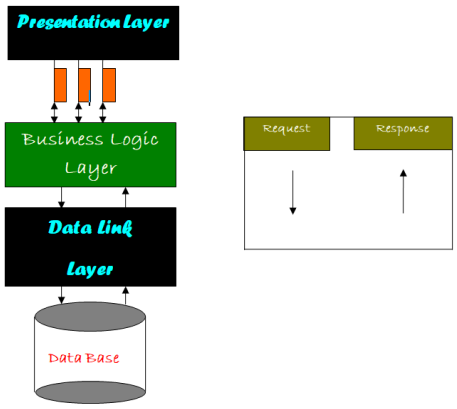


Figure-8 3-Tire Architecture

3. FEASIBILITY STUDY:

The main purpose of the feasibility study is to evaluate the feasibility of Technology, Performance and Economics to add new modules and correct errors in the old operating system. There are features in the preliminary feasibility study section:

- Technological feasibility
- Functionality and feasibility
- Economic viability

3.1 TECHNICAL FEASIBILITY :

The technical problem that is often raised during the feasibility phase of the investigation includes the following:

- Is the necessary technology available to do what is suggested?
- Does the proposed asset have the technical capacity to store the data required for the new system?
- Will the proposed system provide adequate answers to questions, regardless of the number or location of users?
- Can the system be improved if it is upgraded? Are there technical guarantees for accuracy, reliability, easy access and data security?

3.2 OPERATIONAL FEASIBILITY:

3.2.1 User-friendly :

The customer will use the forms in their various activities namely adding new routes, and viewing route details.

3.2.2 Reliability :

The package will download the current activity online.

3.2.3 Security :

The web server and the database server must be protected from hacking, virus etc.

3.2.4 Portability :

The application will be upgraded using standard open-source software (Without Oracle) such as Java, tomcat web server, Internet Explorer Browser etc.

3.2.5 Availability :

This software will always be available.

3.2.6 Maintainability:

A system called wheels uses 2-tier architecture. The first phase is the GUI, which is called the front and the second phase is the website, which uses MySql, in the background.

3.3 ECONOMIC FEASIBILITY

The computer system monitors existing system data flow and processes completely and must generate all manual system reports except a host of other administrative reports. Open-source software such as TOMCAT, JAVA, Mysql and Linux are used to reduce Customer costs.

4. REQUIREMENTS SPECIFICATION:

4.1 FUNCTIONAL REQUIREMENTS SPECIFICATION :

MODULES:

1. ADD RESOURCE

   Applications were uploaded to the site for viewing by users. Resources can be uploaded, modified or deleted.

2. ALLOCATE RESOURCE

   Received applications are viewed by the administrator. Then, manage, find available space and view the app and based on that, the algorithm is used to filter the best user to be assigned or the location to be measured according to the user's quotation sent.

3. USER QUERIES

   Users may have questions about the process. This part of the project is dedicated to making and finding answers to the questions that need to be answered.

4. GRAPH ANALYSIS

Graph analysis is the part where the controller knows the statistics about the data process.

4.2 FUTURE SCOPE OF WORK:

The proposed framework could serve as a first step toward understanding new business models and unlocking the potential of the future EC ecosystem. We are investigating this issue in our ongoing work. Also, integrating the operating costs of ENs into the proposed ME framework is the subject of our future work. Finally, the method of calculating market equity and more complex tasks that capture real-world factors such as labour costs between ENs and data confidentiality is an indication of an interesting future study.

4.3 REQUIREMENT ANALYSIS:

This project involved analyzing the design of a few applications to make the app friendly. To do so, it was really important to keep the navigation from one screen to another in good order and at the same time reduce the amount of typing the user has to perform.

4.4 REQUIREMENT SPECIFICATION:

4.4.1 Functional Requirements:

Graphical User interface with the User.

4.4.2 Software Requirements:

To improve the application the following are the Software Requirements:

1. Python
2. Django
3. Mysql
4. Wampserver

4.4.3 Operating Systems supported:

1. Windows 7
2. Windows XP
3. Windows 8

4.4.4 Technologies and Languages used to Develop:

1. Python

4.4.5 Debugger and Emulator:

- Any Browser (Particularly Chrome)

4.4.6 Hardware Requirements:

To upgrade the application the following are the Hardware requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

5. SYSTEM DESIGN:

5.1 INTRODUCTION:

Systems design: Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

5.2 UML DIAGRAMS:

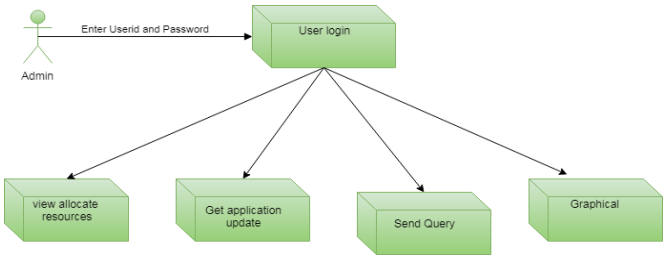
Unified Modeling Language:
The Unified Modeling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic-semantic and pragmatic rules.

5.3 NORMALIZATION:

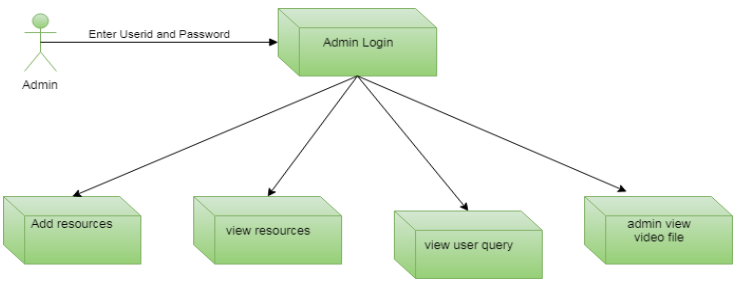
A Database is a collection of interrelated data stored with a minimum of redundancy to serve many applications and is used to group data into several tables and minimizes the artificiality embedded in using separate files.

5.4 SYSTEM DESIGN DIAGRAMS:

1. COMPONENT DIAGRAM
   a. User

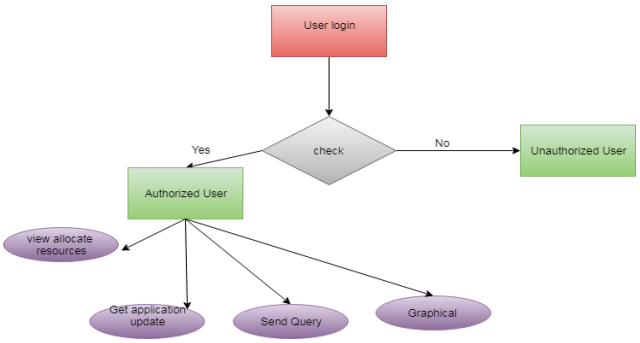


   b. Admin

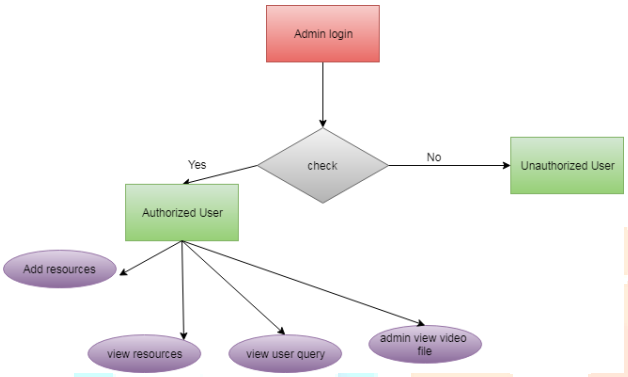


2. ER DIAGRAM

a. User

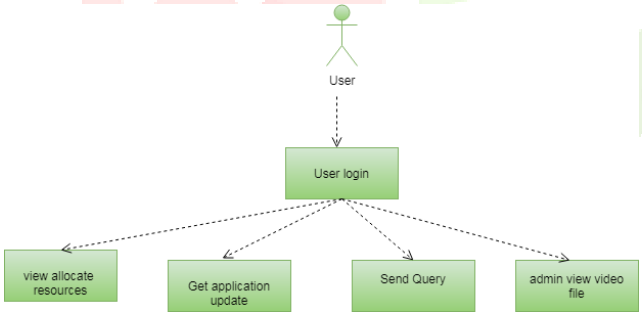


b. Admin

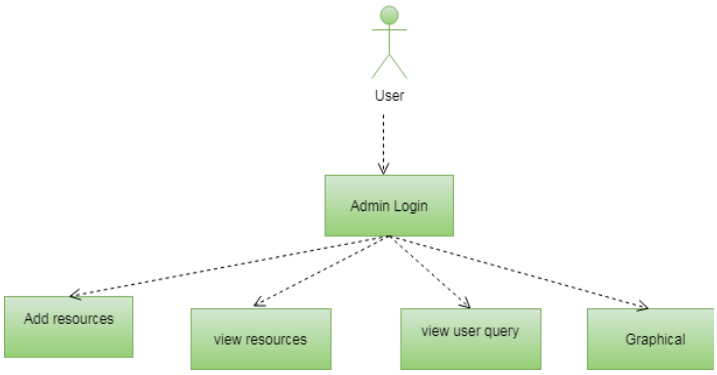


3. USE CASE DIAGRAM

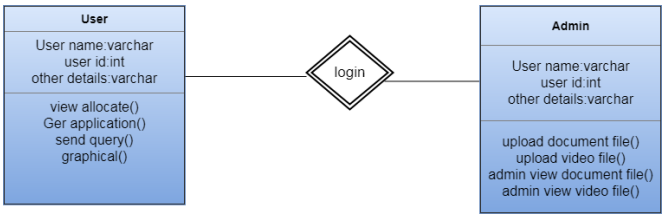
a. User



b. Admin

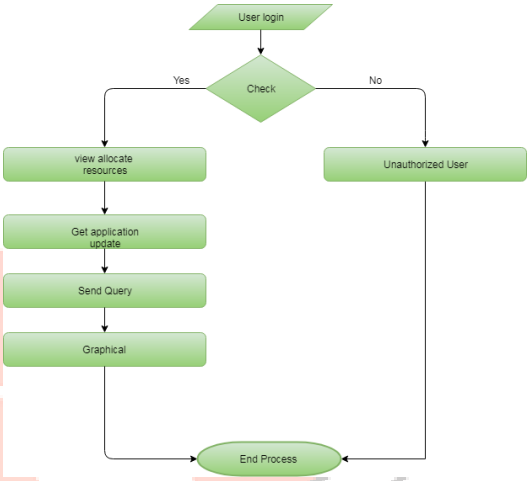


4. CLASS DIAGRAM

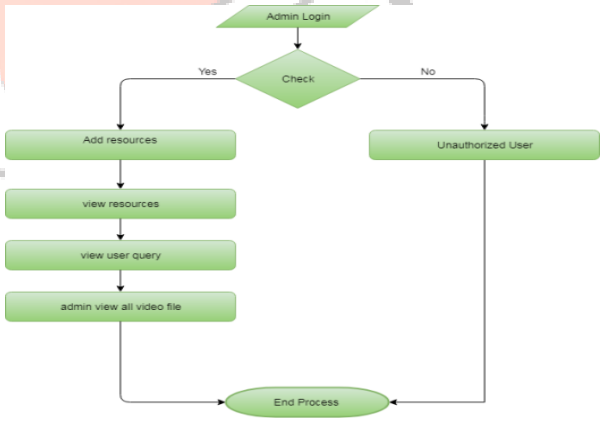


5. DATA FLOW DIAGRAM

a. User

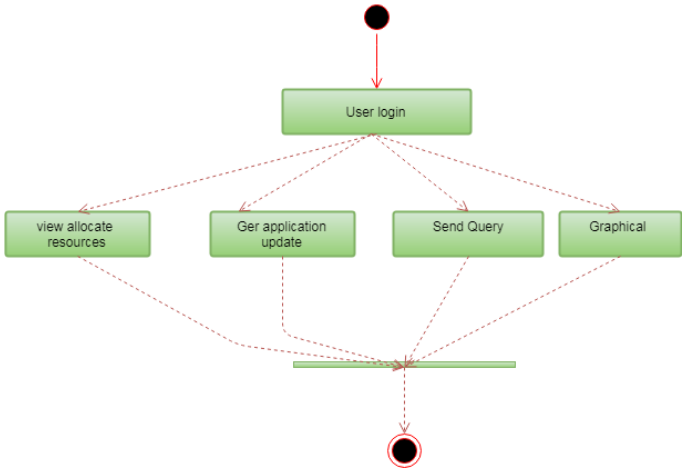


b. Admin

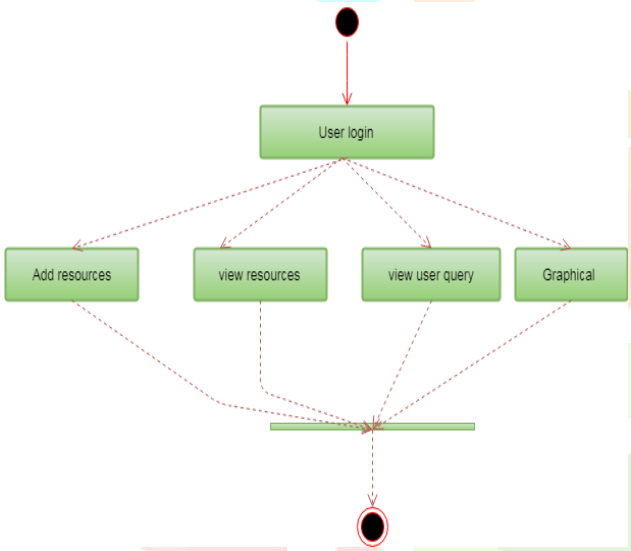


5. ACTIVITY DIAGRAM

a. User

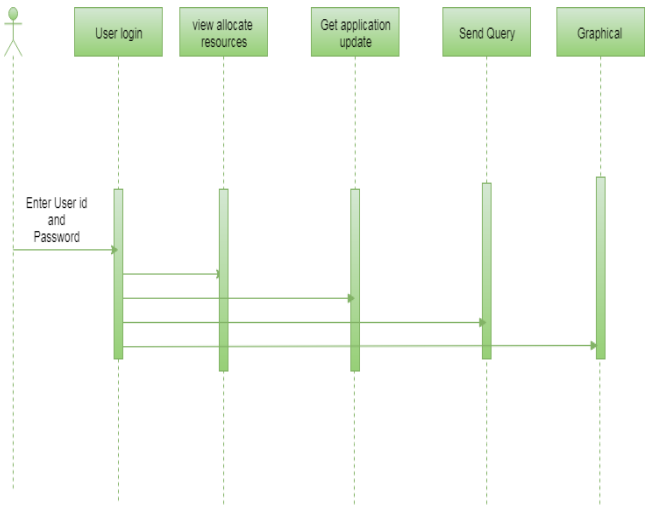


b. Admin

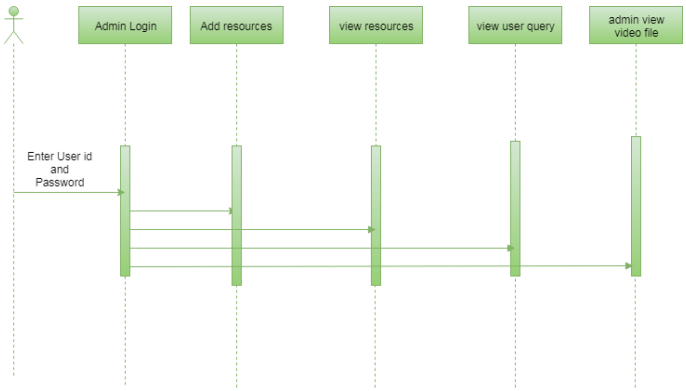


6. SEQUENCE DIAGRAM

a. User



b. Admin



6 IMPLEMENTATION OF CODING:

6.1 TECHNOLOGY USED :

PYTHON

Python is a multi-paradigm editing language. Object-oriented programming and programming are fully supported, and many of its features support functional editing and feature-based programs (including meta-programming and meta-objects (magic modes)).

DJANGO

Django is a free and open-source high-quality Python web framework that promotes rapid development and clean, functional design.

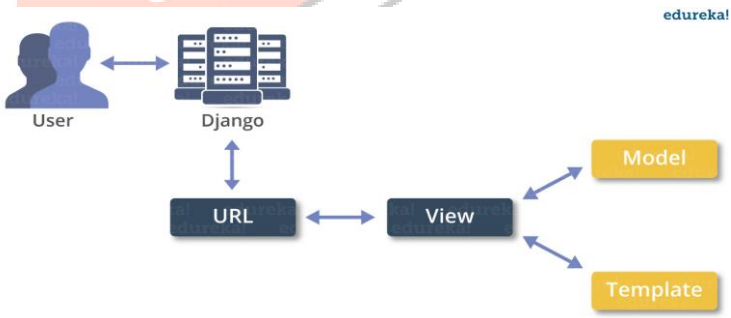


Figure-10 Django Architecture-II

SQLite

SQLite is an embedded SQL data engine. Unlike other SQL websites, SQLite does not have a separate server process.

PyCharm

PyCharm is an integrated development platform (IDE) used for computer programs, especially the Python language. Developed by Czech company JetBrains.

HTML

HTML is not a programming language but is the use of ISO Standard 8879, SGML (Standard Generalized Markup Language), but is specialized in hypertext and familiar with the Web.

JAVA SCRIPT

JavaScript is an integrated, text-based language development software for client and server applications.

6.2 SNIPPETS OF CODE USED:

6.2.1 MANAGE.PY:

The code used to run the server and get the Website in working condition is as follows.

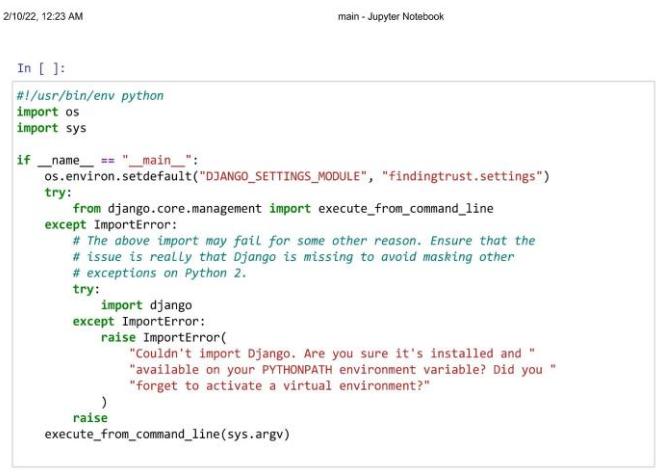


Figure-11 Main Code to run server (manage.py)

7 SYSTEM TESTING:

7.1 INTRODUCTION TO TESTING:

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development.

7.2 TESTING IN STRATEGIES:

To make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

7.2.1 Unit Testing: Unit Testing is done on individual modules as they are completed and become executable. Each module can be tested using the following two Strategies:

7.2.2 Black Box Testing: In this strategy, some test cases are generated as input conditions that fully execute all functional requirements for the program.

7.2.3 White Box testing: In this, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

7.2.4 Integrating Testing: Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated.

7.2.5 System Testing: Involves in-house testing of the entire system before delivery to the user. It aims to satisfy the user the system meets all requirements of the client's specifications.

7.2.6 Acceptance Testing: It is pre-delivery testing in which the entire system is tested at the client's site on real-world data to find errors.

7.3 Test Approach :

Testing can be done in two ways:

7.3.1 Bottom-up Approach: Testing can be performed starting from the smallest and lowest level modules and proceeding one at a time.

7.3.2 Top-down approach: This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written.

7.4 Validation:

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled.

8. Output Screens:

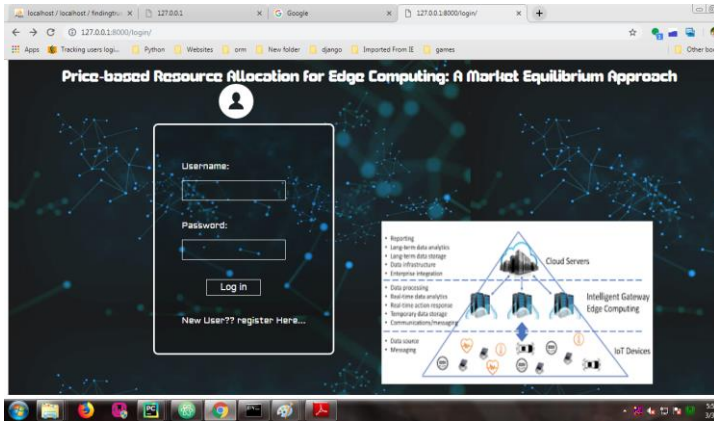


Figure-12 Output Screen 1

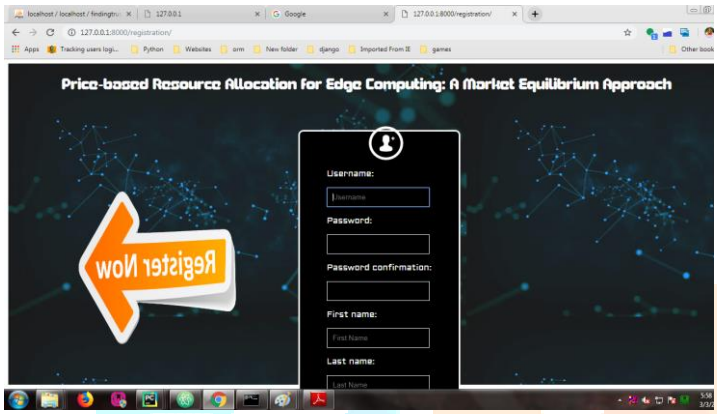


Figure-13 Output Screen 2

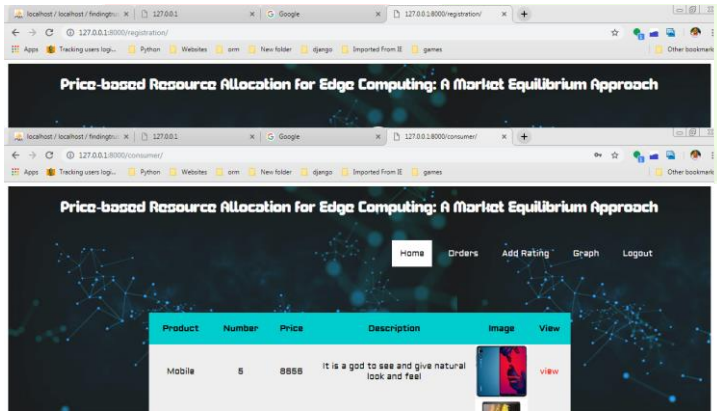


Figure-14 Output Screen 3

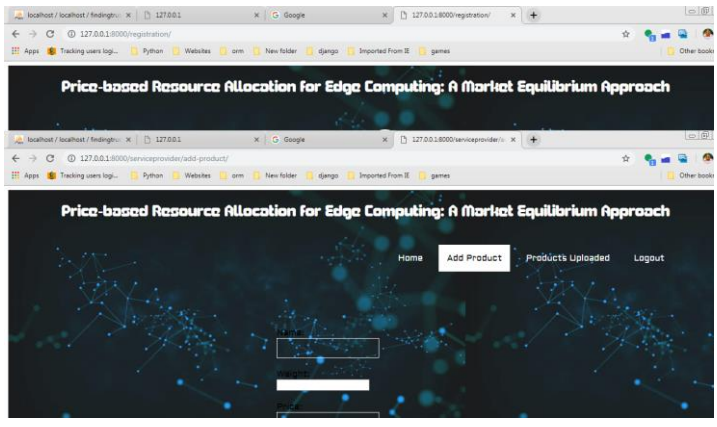


Figure-15 Output Screen 4

9. CONCLUSION:

In this work, we consider the allocation of an EC system consisting of different geographically EN with different programs and a set of services with different desires and purchasing power. Our main contribution is to promote the popular concept of General Equilibrium in Economics as an effective solution to the fundamental problem of EC resources allocation. The proposed solution produces ME which not only works with Pareto but also has many attractive features. The power of this approach greatly exceeds EC applications. For example, it can be used to share the location on the edge with different service providers. We may also use the proposed framework to share resources (e.g., communications, wireless channels) with different users or groups of users (instead of services and service providers). In addition, the proposed model can be extended to a multi-service environment where each customer needs a combination of different service types (e.g., storage, bandwidth, and calculation) to launch their service.

10. 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 an 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.