Amity Journal of Computational Sciences
ISSN 2456-6616 (Online)
Volume 6 Issue 2, 2022

Cloud-based Amizone Assistant

Ayush Sahu
Batchelor of Technology

Sameer
Batchelor of Technology

Vaibhav Srivastava
Batchelor of Technology

Abstract—The cloud-based Amizone Assistant is a virtual aide integrated into the Amizone educational platform. This AI-driven tool aims to enhance educational efficiency and elevate the student experience. Leveraging cloud technology, the assistant offers personalized support, real-time access to academic resources, and streamlined communication. By leveraging the power of the cloud, this innovative solution empowers students and educators alike, fostering a more effective and seamless learning environment.

Keywords—Amizone Assistant, Virtual assistant, AI-driven

• INTRODUCTION

The purpose of this research project is to implement a Telegram bot called AMIZONE Bot that serves as an interactive interface to the AMIZONE web portal. AMIZONE is an online platform widely used by students and faculty members of educational institutions to access important academic information, including attendance records, exam schedules, course details, faculty information, and examination results. The AMIZONE Bot aims to simplify and enhance the user experience by providing a convenient and efficient means of accessing these functionalities directly through the popular messaging platform, Telegram. With the rapid advancement of technology and the increasing prevalence of instant messaging applications, Telegram has emerged as a popular choice for communication and information exchange. By leveraging the capabilities of Telegram, the AMIZONE Bot becomes readily accessible to a large user base, allowing students and faculty members to conveniently interact with the AMIZONE web portal on their mobile devices or computers, without the need for navigating through traditional web browsers

• TECHNOLOGY STACK

A. Python

Python, a widely recognized and versatile programming language, serves as the primary language for developing the AMIZONE Bot. Renowned for its simplicity, readability, and extensive library ecosystem, Python provides an ideal foundation for building chatbots and web applications.

B. Google cloud compute engine

Google Cloud Compute Engine serves as the hosting platform for deploying the AMIZONE Bot. Compute Engine provides scalable and reliable virtual machines (VMs) in the cloud, offering developers the necessary infrastructure to run their applications seamlessly. By leveraging Compute Engine, the AMIZONE Bot can handle multiple user requests concurrently while ensuring stability and optimal performance. Compute Engine's scalability allows the bot to dynamically allocate resources to meet fluctuating user demand. During periods of high usage, the bot can scale up its resources to maintain a responsive user experience. Compute Engine offers features such as load balancing and auto-scaling, optimizing resource allocation and enhancing overall performance. Security is a paramount concern in any application, and Compute Engine addresses this through built-in security measures. Virtual private cloud (VPC) networking, firewalls, and encryption options safeguard the infrastructure and data of the bot.

C. Debian OS

The AMIZONE Bot is deployed on a Compute Engine instance running the Debian operating system (OS). Debian, a popular open-source Linux distribution, is widely recognized for its stability, security, and extensive package repositories.

It provides a reliable and secure foundation for hosting production applications. Debian ensures a secure and customizable environment for executing the bot's code and managing its dependencies. The OS's robust package manager, apt, simplifies the installation and management of software packages, enabling developers to quickly set up the necessary dependencies for the bot's functionality.

In summary, the technology stack employed in the implementation of the AMIZONE Bot showcases the power of modern development and hosting platforms. Python provides flexibility and an extensive library ecosystem for building robust chatbot applications. Google Cloud Compute Engine offers scalability and security for hosting the bot, while Debian OS provides stability and customizability.

- DEPENDENCIES

A. http.client

The http.client library provides functionality for making HTTP requests. It allows the bot to establish connections with the AMIZONE web portal and send HTTP requests to retrieve data such as attendance information, exam schedules, and course details. With http.client, the bot can manage the low-level communication protocols required to interact with the AMIZONE platform.

Requests

The requests library is a widely used Python module that simplifies the process of making HTTP requests. It provides a higher-level interface compared to http.client, making it easier to send GET and POST requests, handle headers, cookies, and parameters, and receive responses from the AMIZONE web portal.

Requests allows the bot to retrieve data from the AMIZONE platform in a more concise and intuitive manner.

B. bs4 (Beautiful Soup)

Beautiful Soup (bs4) is a powerful library for web scraping and parsing HTML. It allows the bot to extract specific data from the HTML structure of the AMIZONE web portal. With Beautiful Soup, the bot can navigate and search through the HTML code, extract relevant information such as attendance details, faculty information, and exam schedules, and present it to the user in a structured format.

Beautiful Soup's parsing capabilities simplify the process of retrieving and manipulating data from the AMIZONE platform.

C. . json

The json module in Python provides functionalities for handling JSON data. JSON (JavaScript Object Notation) is a commonly used data format for exchanging data between systems. In the context of the AMIZONE Bot, JSON is utilized to parse and process data received from the AMIZONE web portal.

The bot can extract relevant information from the JSON responses and present it to the user in a user-friendly format.

D. Telegram

The telegram library is a Python wrapper for the Telegram Bot API, which enables the bot to interact with users on the Telegram messaging platform. With the telegram library, the bot can send and receive messages, reply to user queries, and perform various operations such as sending images, documents, or links.

It provides a convenient interface for integrating the bot with the Telegram platform.

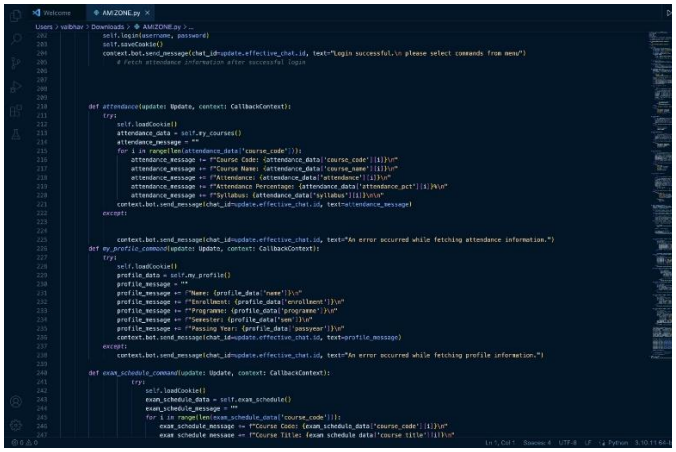
- *IMPLEMENTATION DETAILS*

The AMIZONE Bot project is implemented using a Python class called AMIZONE, which serves as the core component encapsulating the functionalities of interacting with the AMIZONE web portal.

The class provides methods for login, fetching attendance information, accessing profile details, retrieving exam schedules, obtaining course information, retrieving faculty details, and fetching results. In this section, we will delve into the implementation details of the AMIZONE class, discussing its structure, key methods, and how it interacts with the AMIZONE web portal.

A. Class Structure

The class structure in the AMIZONE Bot project plays a crucial role in organizing and encapsulating the functionalities required to interact with the AMIZONE web portal. The class structure provides a modular and organized approach to handle different operations, such as login, fetching attendance, accessing profile details, retrieving exam schedules, obtaining course information, retrieving faculty details, and fetching results. The theory behind the class structure involves the definition of classes and their methods to implement these functionalities effectively.



B. *AMIZONEClass*

The central class in the AMIZONE Bot project is the "AMIZONE" class. This class encapsulates the functionalities related to interacting with the AMIZONE web portal and serves as the main interface for accessing various operations. The class may have instance variables to store information such as session cookies, user credentials, or other necessary data for communication with the AMIZONE portal.

C. *Initialization Method*

The class typically includes an initialization method that is called when an instance of the class is created. This method initializes the necessary attributes and establishes the initial state of the object. For example, it can initialize the session cookies or set up the necessary configurations for communication with the AMIZONE portal.

D. *Login Method*

The login method within the AMIZONE class handles the authentication process. It accepts the user's login credentials as parameters and initiates the login request to the AMIZONE portal. The method may utilize HTTP client functionality or the requests library to communicate with the web portal's login endpoint. Upon successful authentication, the method stores the session cookies to maintain the user's authenticated state for subsequent operations.

E. *Attendance Method*

The attendance method retrieves the attendance information for the user. It sends the necessary HTTP requests to the AMIZONE portal's attendance endpoint and uses web scraping techniques, such as Beautiful Soup, to extract the attendance data from the HTML response. The method may parse the HTML structure to extract relevant details such as subject names, dates, and attendance percentages. The extracted data can be stored in variables or data structures for further processing or presentation.

F. *Profile Method*

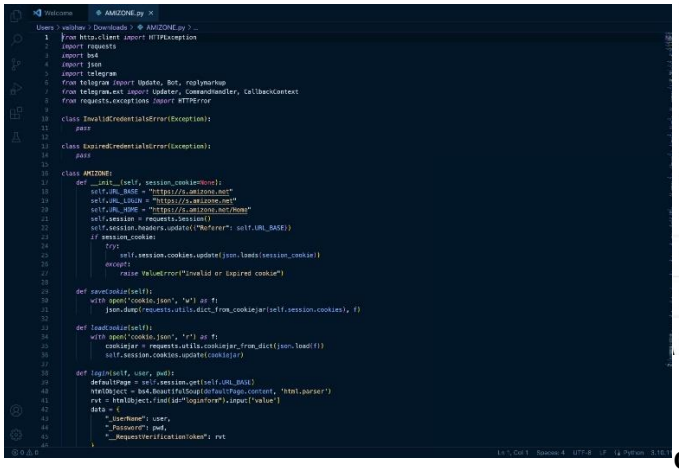
The profile method fetches the user's profile details from the AMIZONE portal. It sends the appropriate HTTP request to the profile endpoint and extracts the relevant information using web scraping techniques. The method can retrieve details such as the user's name, student ID, contact information, and other personal information available in the profile section of the portal. The extracted data can be stored in variables or data structures for later use.

G. Exam Schedule Method

The exam schedule method retrieves the upcoming exam schedules for the user. It sends the necessary HTTP request to the AMIZONE portal's exam schedule endpoint and extracts the schedule information using web scraping techniques. The method can extract details such as subject names, exam dates, timings, and locations. The extracted data can be stored in variables or data structures for further processing or presentation

H. Faculty Details Method

The faculty details method retrieves information about the faculty members associated with the user's courses. It sends the necessary HTTP request to the faculty details endpoint and extracts the relevant data using web scraping techniques. The method can extract details such as faculty names, qualifications, contact information, and office hours. The extracted data can be stored in variables or data structures for further processing or presentation.



V. *DEPLOYMENT ON GOOGLE CLOUD COMPUTE ENGINE*

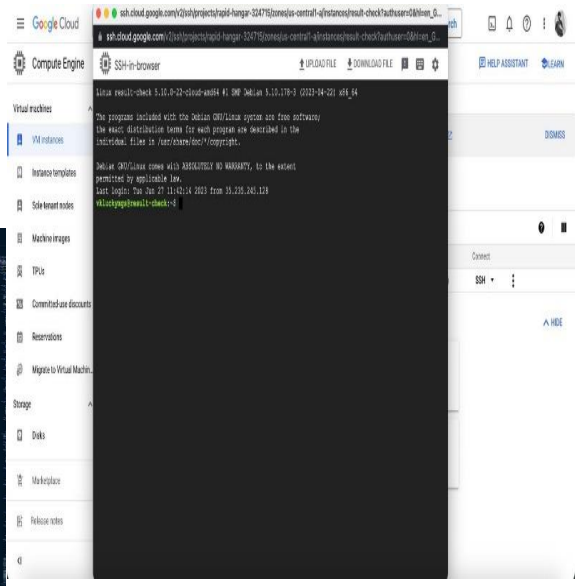
To deploy the AMIZONE Bot on Google Cloud Compute Engine with Debian OS Deployment is a critical phase in software development where the application or system is made available for use.

In the case of the AMIZONE Bot, deploying it on Google Cloud Compute Engine offers a reliable and scalable hosting platform. This section will delve into the theory behind deploying the AMIZONE Bot on Google Cloud Compute Engine with Debian OS. Google Cloud Compute Engine is an infrastructure-as-a-service (IaaS) offering from Google Cloud Platform. It allows users to create and manage virtual machines in the cloud, providing scalable computing resources for hosting applications and services. Deploying the AMIZONE Bot on Compute Engine offers several benefits, including reliability, scalability, security, and ease of management

Setting up a Virtual Machine :

To deploy the AMIZONE Bot on Google Cloud Compute Engine, the first step is to create a virtual machine (VM) instance.

The VM acts as the host for the bot and provides the necessary computing resources to run the application. During the VM creation process, users can select the desired machine type, CPU, memory, and other configuration options based on their requirements.



Choosing the Operating System:

For deploying the AMIZONE Bot on Compute Engine, the Debian operating system is chosen. Debian is a popular Linux distribution known for its stability, security, and extensive package ecosystem.

It provides a reliable and well-maintained environment for running applications. Google Cloud Compute Engine offers a wide range of operating system options, and Debian is a suitable choice for deploying Python-based applications.

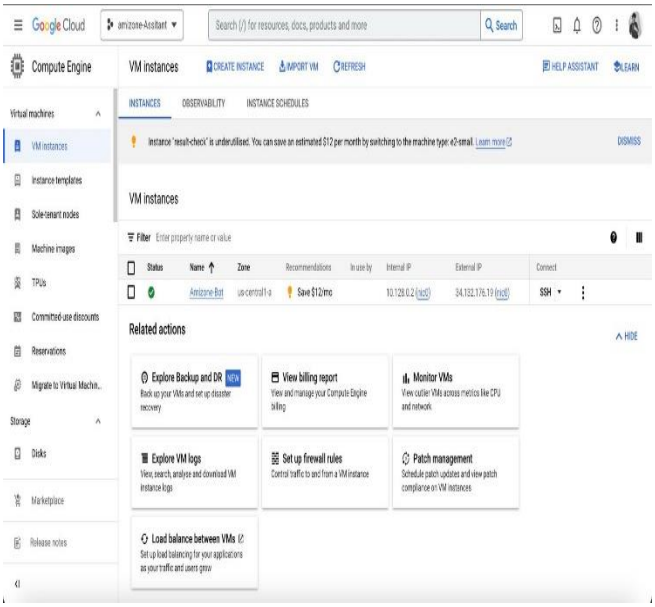
Installing Dependencies:

Once the VM instance with Debian OS is set up, the next step is to install the necessary dependencies for running the AMIZONE Bot.

This includes installing Python, the required Python libraries, and any other system-level dependencies. Using package managers like `apt` or `pip`, the dependencies can be installed easily on the Debian OS.

Transferring the Bot Code:

To deploy the AMIZONE Bot on the Compute Engine VM, the bot's code needs to be transferred to the VM instance. This can be done using various methods such as SSH, SCP, or utilizing Google Cloud Storage for efficient file transfer. Once the code is transferred to the VM, it can be stored in the appropriate directory.

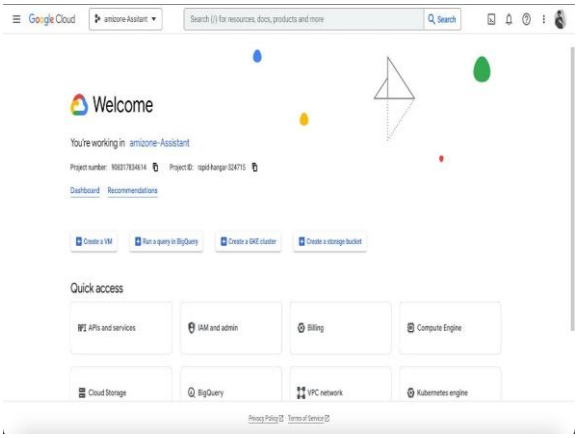


Configuring Firewall and Networking:

To ensure that the AMIZONE Bot is accessible over the network, appropriate firewall and networking configurations need to be set up. Compute Engine provides firewall rules that allow users to define inbound and outbound traffic rules.

Users can configure firewall rules to allow incoming traffic on the desired ports required for the bot's operation.

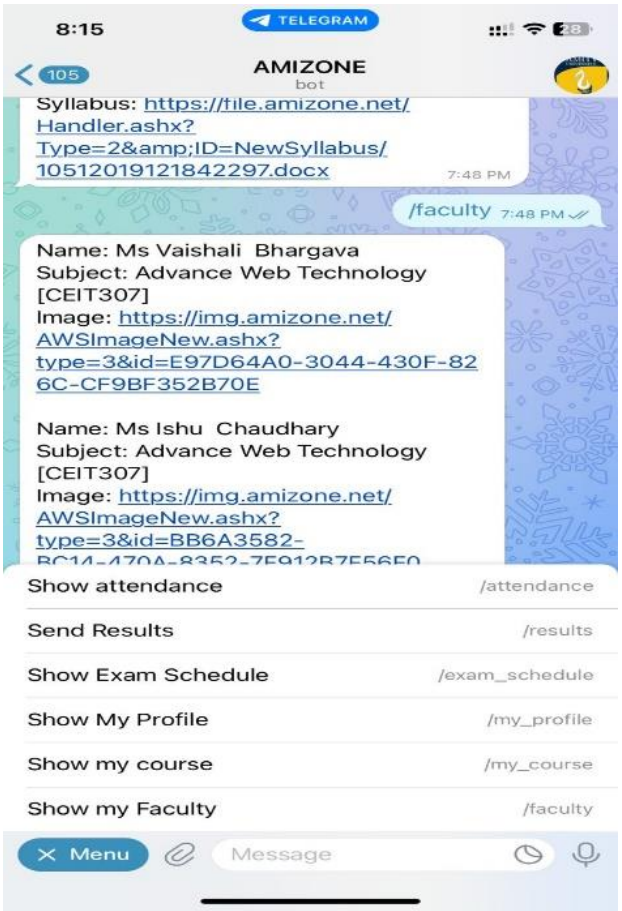
Additionally, networking settings such as IP addresses and DNS configurations can be set up to provide convenient access to the bot.



Running the Bot:

With the dependencies installed, the code transferred, and the necessary configurations in place, the AMIZONE Bot can be launched on the Compute Engine VM. This typically involves running the Python script or executing the necessary commands to start the bot's execution.

The bot can be set up as a background process or managed using process managers like `supervisor` or `systemd` to ensure its availability and reliability.



Security Considerations:

When deploying the AMIZONE Bot on Compute Engine, it is crucial to address security considerations. This includes configuring secure communication channels (HTTPS), ensuring the VM instance has the necessary security patches and updates, and employing best practices for authentication and access control.

Compute Engine also offers additional security features like firewall rules, network segmentation, and identity and access management (IAM) to enhance the security posture of the deployed bot.

Backup and Disaster Recovery:

To protect against data loss or system failures, regular backups and disaster recovery strategies should be implemented. Compute Engine provides options for creating snapshots or using managed backup services to safeguard the bot's data and configuration.

It is essential to establish backup schedules and procedures to ensure data integrity and minimize downtime in case of any unforeseen incidents. In conclusion, deploying the AMIZONE Bot on Google Cloud Compute Engine with Debian OS provides a robust and scalable infrastructure for hosting the application. Leveraging the VM instances, networking capabilities, and monitoring tools offered by Compute Engine, users can ensure the availability, performance, and security of the bot.

By following best practices in deployment, security, and maintenance, the AMIZONE Bot can provide a seamless and reliable experience to users accessing its functionalities through Telegram messages.

VI. *CONCLUSION*

The implementation of the AMIZONE Bot provides a convenient way for users to interact with the AMIZONE web portal through Telegram.

Users can easily access their attendance information, profile details, exam schedules, course information, faculty details, and results by sending commands to the bot.

The AMIZONE Bot serves as a bridge between users and the AMIZONE web portal, simplifying the process of retrieving essential information and performing various tasks related to academic activities.

By leveraging the capabilities of Telegram and integrating with the AMIZONE portal, the bot offers a seamless and user-friendly experience for students, faculty, and other stakeholders.

One of the key advantages of the AMIZONE Bot is its accessibility. Telegram is a widely used messaging platform that is available on various devices and operating systems, making it easily accessible to a large user base. Users can interact with the bot from their smartphones, tablets, or computers, eliminating the need to log in to the AMIZONE web portal through traditional means. This convenience allows users to stay updated on their academic progress and access relevant information on the go.

The bot's command-based interface simplifies the process of retrieving information from the AMIZONE web portal. Instead of navigating through multiple web pages and menus, users can simply send commands to the bot and receive the desired information directly in their Telegram chat.

Another significant functionality of the AMIZONE Bot is retrieving exam schedules. Users can easily obtain information about upcoming exams, including the date, time, and venue. This feature helps users plan their study schedule and stay organized, ensuring they are well prepared for their examinations. Course information is also readily available through the bot. Users can fetch details about their enrolled courses, including course codes, titles, and instructors. This functionality provides users with a quick overview of their course load and helps them stay informed about their academic curriculum.

a. Convenience: The bot provides a convenient and user-friendly interface for accessing AMIZONE services. Users can interact with the bot through natural language commands, eliminating the need to navigate complex web pages or menus.

b. Time-saving: The bot automates the retrieval of information from the AMIZONE portal, saving users time and effort. They can quickly access attendance records, exam schedules, course details, and other information without manually searching for it on the web portal.

c. Real-time updates: The bot's web scraping capabilities

ensure that users receive real-time updates and the latest information from the AMIZONE portal. This eliminates the need for users to constantly check the portal for updates or changes.

d. Personalization: The bot allows users to access personalized information by maintaining session cookies. Users can stay logged in and retrieve their individualized data without the need to provide login credentials for each interaction.

e. Seamless integration: By integrating with the Telegram messaging platform, the bot seamlessly integrates into users' existing communication channels. Users can access AMIZONE services through a platform they are already familiar with, enhancing user adoption and reducing the learning curve.

f. Extensibility: The bot's architecture and API integration enable future enhancements and integration with additional services. It can be extended to include features like reminders, notifications, or integration with other educational platforms, further enriching the user experience.

g. Accessibility: The bot's deployment on a platform like Google Cloud Compute Engine ensures continuous availability and global accessibility. Users can access the bot at any time, from anywhere, using their preferred device.

In conclusion, the AMIZONE Bot project combines web scraping, API integration, and bot development techniques to enhance the user experience and provide easy access to AMIZONE services. By leveraging these technologies, the project offers convenience, timesaving benefits, real-time updates, personalization, seamless integration, extensibility, and global accessibility. The project showcases the power of technology in simplifying and improving academic management processes, ultimately benefiting students, faculty, and other stakeholders involved in the AMIZONE.

VII . Reference

[1] P. K. Kushwaha and M. Kumaresan, "Machine learning algorithm in healthcare system: A Review," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 478-481, doi: 10.1109/ICTAI53825.2021.9673220.

[2] P. K. Kushwaha, V. Bibhu, B. P. Lohani and D. Singh, "Review on information security, laws and ethical issues with online financial system," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 49-53, doi: 10.1109/ICICCS.2016.7542350.

[3] G. Gulati, B. P. Lohani and P. K. Kushwaha, “A Novel Application Of IoT In Empowering Women Safety Using GPS Tracking Module,” 2020 Research, Innovation, Knowledge Management and Technology Application for Business Sustainability (INBUSH), Greater Noida, India, 2020, pp. 131-137, doi: 10.1109/INBUSH46973.2020.9392193.

[4] D. Pareta, I. N. Verma, B. P. Lohani, P. K. Kushwaha and V. Bibhu, "IoT Enabled Smart and Efficient Musical Water Fountain," 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India, 2022, pp. 369-373, doi: 10.1109/ICIPTM54933.2022.9754129.

[5] B. P. Lohani, M. Trivedi, R. J. Singh, V. Bibhu, S. Ranjan and P. K. Kushwaha, "Machine Learning Based Model for Prediction of Loan Approval," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 465-470, doi: 10.1109/ICIEM54221.2022.9853160

[6] V. Bibhu, A. Kumar, B. P. Lohani and P. K. Kushwaha, "Robust Secured Framework for Online Business Transactions over Public Network," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 555-560, doi: 10.1109/ICIEM51511.2021.9445380.

[7] V. Bibhu, P. K. Kushwaha and B. P. Lohani, "A review of security of the cloud computing over business with implementation," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Greater Noida, India, 2016, pp. 192-198, doi: 10.1109/ICICCS.2016.7542342.

[8] Amardeep Gupta and Ranjeet Kumar Rout, “ROTEE: Remora Optimization and Tunicate swarm algorithm-based Energy-Efficient cluster-based routing for EHenabled heterogeneous WSNs,” International Journal of Communication System (Q1), vol. 33, no. 6, pp. 1-23, 2022. DOI:https://doi.org/10.1002/dac.5372. (IF: 2.0)

[9] A. D. Gupta and R. K. Rout, "An Effective Optimization Method for Energy Efficient Clustering in EH Wireless Sensor Networks," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 699-702, doi: 10.1109/ICTAI53825.2021.9673312.

[10] S. S. N. Challapalli, P. Kaushik, S. Suman, B. D. Shivahare, V. Bibhu and A. D. Gupta, "Web Development and performance comparison of Web Development Technologies in Node.js and Python," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 303-307, doi: 10.1109/ICTAI53825.2021.9673464.

[11] A. D. Gupta, S. Suman, S. S. N. Challapalli, P. Kaushik and V. Bibhu, "Survey Paper: Comparative Study of Machine Learning Techniques and its Recent Applications," 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India, 2022, pp. 449-454, doi: 10.1109/ICIPTM54933.2022.9754206.

[12] Ahlawat, A., Rana, A., Goyal, N. et al. Potential role of nitric oxide synthase isoforms in pathophysiology of neuropathic pain. Inflammopharmacol 22, 269–278 (2014). https://doi.org/10.1007/s10787-014-0213-0

[13] A. R. Yeruva, P. Choudhari, A. Shrivastava, D. Verma, S. Shaw and A. Rana, "Covid-19 Disease Detection using Chest X-Ray Images by Means of CNN," 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 2022, pp. 625-631, doi: 10.1109/ICTACS56270.2022.9988148.

[14] Ghosh, S., Rana, A., & Kansal, V. (2020). A benchmarking framework using nonlinear manifold detection techniques for software defect prediction. International Journal of Computational Science and Engineering, 21(4), 593-614.

[15] Raghavendra, M. S., Chawla, P., & Rana, A. (2020, June). A survey of optimization algorithms for fog computing service placement. In 2020 8th international conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO) (pp. 259-262). IEEE.

[16] Gupta, S., Rana, A., & Kansal, V. (2020). Optimization in wireless sensor network using soft computing. In Proceedings of the Third International Conference on Computational Intelligence and Informatics: ICCII 2018 (pp. 801-810). Springer Singapore.

[17] Kunwar, V., Agarwal, N., Rana, A., & Pandey, J. P. (2018). Load balancing in cloud—a systematic review. Big Data Analytics: Proceedings of CSI 2015, 583-593.
[18] Chawla, P., Chana, I., & Rana, A. (2015). A novel strategy for automatic test data generation using soft computing technique. Frontiers of Computer Science, 9, 346-363.
[19] Walia, H., Rana, A., & Kansal, V. (2017, September). A Naïve Bayes Approach for working on Gurmukhi Word Sense Disambiguation. In 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 432-435). IEEE.
[20] Dash, Y., Dubey, S. K., & Rana, A. (2012). Maintainability prediction of object oriented software system by using artificial neural network approach. International Journal of Soft Computing and Engineering (IJSCE), 2(2), 420-423.
[21] Dubey, S. K., & Rana, A. (2010). A comprehensive assessment of object-oriented software systems using metrics approach. International Journal on Computer Science and Engineering, 2(8), 2726-2730.
[22] S. Gupta, A. Rana and V. Kansal, "Comparison of Heuristic techniques:A case of TSP," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 172-177, doi: 10.1109/Confluence47617.2020.9058211.
[23] Ghosh, S., Rana, A., & Kansal, V. (2018). A nonlinear manifold detection based model for software defect prediction. Procedia computer science, 132, 581-594.
[24] Chawla, P., Chana, I., & Rana, A. (2016). Cloud-based automatic test data generation framework. Journal of Computer and System Sciences, 82(5), 712-738.
[25] Bhardwaj, M., & Rana, A. (2016). Key Software Metrics and its Impact on each other for Software Development Projects. International Journal of Electrical & Computer Engineering (2088-8708), 6(1).
[26] Rana, A., & Sharma, S. (2016). Mechanism of sphingosine-1-phosphate induced cardioprotection against I/R injury in diabetic rat heart: Possible involvement of glycogen synthase kinase 3β and mitochondrial permeability transition pore. Clinical and Experimental Pharmacology and Physiology, 43(2), 166-173.
[27] G. Dubey, A. Rana and N. K. Shukla, "User reviews data analysis using opinion mining on web," 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), Greater Noida, India, 2015, pp. 603-612, doi: 10.1109/ABLAZE.2015.7154934.
[28] Ghosh, S., Rana, A., Kansal, V. (2017). Predicting Defect of Software System. In: Satapathy, S., Bhateja, V., Udgata, S., Pattnaik, P. (eds) Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications . Advances in Intelligent Systems and Computing, vol 516. Springer, Singapore. https://doi.org/10.1007/978-981-10-3156-4_6
[29] Sanjay Kumar Dubey, Ajay Rana, and Yajnaseni Dash. 2012. Maintainability prediction of object-oriented software system by multilayer perceptron model. SIGSOFT Softw. Eng. Notes 37, 5 (September 2012), 1–4. https://doi.org/10.1145/2347696.2347703
[30] S. Chawla, G. Dubey and A. Rana, "Product opinion mining using sentiment analysis on smartphone reviews," 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2017, pp. 377-383, doi: 10.1109/ICRITO.2017.8342455.
[31] Dubey, S. K., Rana, A., & Sharma, A. (2012). Usability evaluation of object oriented software system using fuzzy logic approach. International Journal of Computer Applications, 43(19), 1-6.
[32] Saini, Rimmi, Sanjay Kumar Dubey, and Ajay Rana. "Analytical study of maintainability models for quality evaluation." Indian Journal of Computer Science and Engineering 2.3 (2011): 449-454.
[33] Ghosh, Soumi, Ajay Rana, and Vineet Kansal. "A statistical comparison for evaluating the effectiveness of linear and nonlinear manifold detection techniques for software defect prediction." International Journal of Advanced Intelligence Paradigms 12.3-4 (2019): 370-391.
[34] A. Singh, M. Chaudhary, A. Rana and G. Dubey, "Online Mining of data to generate association rule mining in large databases," 2011 International Conference on Recent Trends in Information Systems, Kolkata, India, 2011, pp. 126-131, doi: 10.1109/ReTIS.2011.6146853.
[35] N. Tyagi, A. Rana and V. Kansal, "Creating Elasticity with Enhanced Weighted Optimization Load Balancing Algorithm in Cloud Computing," 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 2019, pp. 600-604, doi: 10.1109/AICAI.2019.8701375.
[36] Dubey, Sanjay Kumar, and Ajay Rana. "A fuzzy approach for evaluation of maintainability of object oriented software system." International Journal of Computer Applications 49.21 (2012).