FMDB *Transactions on Sustainable Intelligent Networks*

VehiWalnut: Smart Fuel Management with Real-Time Data and User Interaction

R. Regin[1, *], R. S. Gaayathri[2], P. Paramasivan[3], S. Suman Rajest[4], Radwa Radwan[5]

[1]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[2]Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[3]Department of Research and Development, Dhaanish Ahmed College of Engineering, Chennai, Tamil Nadu, India.
[4]Department of Research and Development & International Student Affairs, Dhaanish Ahmed College of Engineering, Chennai, Tamil Nadu, India.
[5]Department of Business Administration, College of Humanities and Administration Studies, Onaizah Colleges, Unaizah, Saudi Arabia.
[5]Department of Business Administration, Higher Future Institute for Specialized Technological Studies, Future Academy, Obour City, Egypt.
regin12006@yahoo.co.in[1], gr4829@srmist.edu.in[2], paramasivanchem@gmail.com[3], sumanrajest414@gmail.com[4], radwa.mohamed@fa-hists.edu.eg[5]

Abstract: This research paper examines the creation and execution of VehiWalnut, a cutting-edge system developed to enhance fuel management processes. VehiWalnut incorporates cutting-edge technologies such as data visualization, cloud-based chatbot evaluation, and deep learning to improve efficiency and user satisfaction. A thorough literature review is conducted to identify important studies in data visualization, Web API analysis, chatbot performance evaluation, and deep learning techniques for machine learning chatbots. These studies are then combined to form the basis of the research. The text describes the development of VehiWalnut, emphasizing its diverse functionalities and the use of the Python programming language for its implementation. The system allows users to enter fuel fill-up data through WhatsApp messages, which are processed using regular expressions to extract and insert the data into a SQLite database. VehiWalnut allows users to request detailed reports on fuel expenses, timing of fuel refills, predictions for the next refill date, and other related information. The system's effectiveness is assessed by conducting case studies and performance evaluations, showcasing its capacity to enhance fuel management procedures for individual and commercial users. The research findings offer valuable insights into incorporating advanced technologies to improve fuel management systems, thereby paving the way for future advancements in this domain.

Keywords: Fuel Management; WhatsApp Chatbot; Data Visualization; Machine Learning; Predictive Analysis; Python Programming; Automated Reporting; Twilio API; SQLite database.

Received on: 13/09/2023, **Revised on:** 29/11/2023, **Accepted on:** 19/12/2023, **Published on:** 05/03/2024

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIN

DOI: https://doi.org/10.69888/FTSIN.2024.000157

Cite as: R. Regin, R. S. Gaayathri, P. Paramasivan, S. S. Rajest, and R. Radwan, "VehiWalnut: Smart Fuel Management with Real-Time Data and User Interaction," *FMDB Transactions on Sustainable Intelligent Networks*., vol.1, no.1, pp. 56–71, 2024.

1. Introduction

*Corresponding author.

Effective fuel management is essential for private vehicle owners and commercial fleets because it directly impacts operating costs, environmental sustainability, and overall vehicle efficiency [11]. Conventional techniques for keeping tabs on fuel consumption and costs frequently rely on labor-intensive, error-prone manual documentation. Advanced systems that use contemporary technologies have shown promise as answers to these problems [12]. This paper introduces VehiWalnut, an advanced fuel management system that combines machine learning, cloud-based chatbot performance evaluation, and data visualization to improve fuel tracking and management. Over the past few decades, there have been significant advancements in fuel management systems [13].

Fuel costs and consumption were first tracked manually using calculations and basic logbooks. However, more sophisticated solutions have been created since the introduction of digital technologies [14]. These days, fuel management systems frequently include real-time data analytics, automated reporting, and predictive maintenance features. Despite these advancements, many existing systems still have issues with scalability, accuracy of data, and user engagement [15]. Manual data entry, which can be laborious and prone to human error, is one of the main problems with conventional fuel management techniques. Users of manual processes must keep track of mileage, fuel purchases, and other pertinent data, which frequently results in inaccurate or incomplete data entries [16]. These errors can impair the efficacy of fuel management plans and make it more difficult to make defensible choices [17].

VehiWalnut uses cutting-edge technology to improve user experience, expedite data entry, and boost system performance to solve these problems. Real-time data entry and communication via well-known messaging services like WhatsApp are made possible by integrating a cloud-based chatbot with multiple APIs [18]. This integration improves user accessibility and streamlines the data entry process, which raises user engagement levels. The increasing ubiquity of mobile messaging apps like WhatsApp and other platforms opens up new possibilities for improving fuel management systems [19]. With its widely used interface and ease of use, WhatsApp provides a convenient way for users to enter data and get updates instantly. VehiWalnut enhances user engagement and data entry efficiency by integrating chatbot technology with WhatsApp [20]. This leads to better accessibility and fuel management efficiency. With advanced machine learning algorithms and natural language processing (NLP) techniques, VehiWalnut's chatbot can precisely comprehend and react to user inquiries [21]. With its smooth API integration, this chatbot can handle natural language interactions and offer individualized help and up-to-date information [22].

A more natural and intuitive user experience is ensured by the chatbot's ability to learn from user interactions and refine its responses over time to the application of machine learning algorithms. SQLite is a lightweight, serverless database that powers the system's backend and makes data management easier [23]. Thanks to SQLite's flexibility, VehiWalnut can handle massive amounts of data without complicated infrastructure, making it appropriate for small fleet managers and individual car owners. VehiWalnut gives users a thorough understanding of fuel consumption patterns by storing pertinent data such as fuel type, the amount paid, vehicle name, kilometers driven, and fill-up date and time [24]. VehiWalnut integrates several Python libraries for machine learning, data manipulation, and visualization to improve user experience and system functionality further. With libraries like Pandas, Matplotlib, and Scikit-learn, the system can efficiently process and analyze large datasets, extract insightful information, and present the results in an understandable format [25]. For instance, Matplotlib produces visualizations that enable users to rapidly understand fuel consumption patterns and make wise decisions regarding their driving practices. User engagement is greatly increased by VehiWalnut's integration with Twilio's API for WhatsApp communication [26].

Twilio's API expands the system's audience by enabling users to engage with it via a recognizable messaging app, making it accessible to people who might not feel comfortable utilizing more conventional web applications. This feature streamlines the feedback and data entry processes, increasing user convenience and effectiveness [27]. VehiWalnut's dependability and efficiency were validated by the project's performance analysis, which included compile and run times and machine learning accuracy. High accuracy in fuel type prediction using a Random Forest Classifier based on input data is essential to preserving user satisfaction and trust [28]. Tools for data visualization, such as confusion matrices and graphs showing compile and run times, gave precise insights into system performance and model accuracy [29]. Despite its many advantages, the VehiWalnut project encountered difficulties with data consistency and quality, user adoption and training, integration complexity, scalability, and performance. It was difficult to ensure user-entered data was accurate and consistent because incomplete or inaccurate data made it harder for the system to provide precise insights and recommendations [30].

The system includes validation checks and prompts users to fix any errors during data entry to address this. Getting users to adopt and use the system regularly took a lot of work. It's possible that some users won't adapt well to change or won't see the system's benefits right away [31]. To help users overcome this challenge, the project team concentrated on creating an intuitive user interface and offering support and guidance. One of the technical challenges was integrating the chatbot with multiple APIs and ensuring that different system components were communicating smoothly [32]. The development team closely monitored API requests and responses to keep the system responsive and reliable. Ensuring the system's performance and scalability as the user base and data volume grew became crucial. To solve these problems, the group used various optimization techniques, such as database indexing and query performance optimization [33].

The VehiWalnut project lays a strong basis for upcoming enhancements and developments. Improved machine learning models, telematics system integration, user feedback and customization, expanding platform compatibility, and sophisticated analytics and reporting could be the main areas of future work [34]. The VehiWalnut system's current data analysis capabilities may be expanded by adding more advanced analytics and reporting features. On a monthly or quarterly basis, for example, users may receive comprehensive reports on their fuel consumption that provide insightful information and highlight areas for improvement [35]. Fleet managers may obtain additional information by integrating the VehiWalnut system with telematics platforms. This information, including vehicle location, speed, and driving style, could lead to more in-depth analysis and deeper insights into operations [36].

Future research projects might use deep learning techniques or other advanced machine learning models to improve the system's predictive power. These models could provide more accurate and customized recommendations by analyzing a greater range of fuel-related factors [37]. Maintaining the system's success will depend on gathering user feedback and incorporating it into future versions. Offering customization options, like setting fuel consumption goals or receiving alerts for unusual fuel usage patterns, could improve the system's value and user satisfaction [38]. The VehiWalnut system's accessibility and reach may be improved by adding more messaging services and mobile apps to its list of compatible platforms. This expansion might entail creating specialized iOS and Android mobile apps and integrating WeChat and Telegram platforms [39]. In conclusion, the VehiWalnut project successfully demonstrates the possibilities of fusing databases, application programming interfaces (APIs), machine learning techniques, and contemporary web development frameworks to produce an intelligent fuel management system [40].

The project's value to fleet managers and vehicle owners is highlighted by its accomplishments in improving fuel efficiency, increasing user engagement, and providing accurate predictions and recommendations. Notwithstanding the difficulties encountered, the VehiWalnut system has shown to be a dependable and scalable solution with ample opportunity for further advancements. As the project develops, it is expected to yield additional benefits and support fuel management strategies that are more sustainable and effective.

2. Literature Review

Gupta and Bagchi [1] emphasize the essential role of data visualization in transforming complex data into understandable formats. They explain that visual tools such as graphs and maps are instrumental in uncovering patterns and trends that remain obscured in raw or tabular data. The chapter provides a comprehensive overview of various methods for visualizing data using Python, demonstrating how these techniques can significantly enhance data comprehension and analysis. By leveraging Python's powerful libraries, data scientists and non-experts can gain deeper insights into their data, making informed decisions and effectively communicating findings to a broader audience.

Serbout et al. [2] conducted an extensive study analyzing 42,194 OpenAPI Specifications (OAS) from GitHub to gain insights into the size and diversity of Web APIs. The study revealed that most APIs are relatively small, highlighting a medium correlation between the size of APIs' functional structures and their data models. This correlation suggests a balanced development approach where functional and data components grow together. Additionally, the study found that developers frequently reuse schema definitions within the same API model, indicating a trend toward efficiency and consistency in API design. These findings provide a comprehensive understanding of the current landscape of Web APIs.

Gunnam et al. [3] investigate performance assessment methods for cloud-based chatbot systems in heterogeneous environments. Key metrics include response time, throughput, and load testing. Evaluating both real and simulated users, the study provides insights into chatbot performance, aiding practitioners in development and deployment decisions.

Alia et al. [4] emphasize the widespread use of WhatsApp for business communication due to its convenience, with 92.1% of internet users aged 16 to 64 utilizing the platform. Virtual meetings between buyers and sellers are crucial, prompting the adoption of chatbots to handle repetitive inquiries efficiently. The study assesses chatbot effectiveness through the Waterfall System Development Life Cycle (SDLC) and User Acceptance Testing (UAT). Questionnaire results affirm high satisfaction, indicating the feasibility and effectiveness of chatbot implementation in providing essential customer information.

Singh [5] introduces a method for generating back-channel responses in text-based chats to stimulate discussions. Utilizing Python, the Chatbot system implemented via the Telegram application facilitates direct and widespread information communication. Back-channel comments help the Chatbot engage as a conversation participant, eliciting more attentive user responses. The study highlights the effectiveness of this approach, showing improved conversation activation by posting both back-channel queries and replies triggered by question marks in remarks.

Attigeri et al. [6] examine the necessity for quality education by facilitating prospective students' acquisition of university information. They advocate for chatbot applications on university websites to answer student queries accurately, leveraging advanced NLP models. The study compares five chatbot models, finding neural network-based approaches outperform TF-IDF and pattern-matching methods, with sequential modeling showing superior accuracy and mitigating over-fitting concerns.

Ali and Ali [7] examined deep learning approaches for machine learning chatbots, evaluating their effectiveness in response generation quality, accuracy, and user satisfaction. The study compares models such as recurrent neural networks and transformers, revealing promising performance. However, challenges persist regarding data availability, model complexity, and scalability. This meta-analysis guides researchers and practitioners in selecting and optimizing deep learning approaches to enhance intelligent conversational agents.

Lin and Yu [8] review AI chatbot applications in education using bibliometric and citation network analysis. Their findings show that post-COVID-19, chatbot use in education has surged, focusing on technological advancements, student perceptions, and effectiveness. Key applications include language education, educational services, and healthcare training. Despite diverse contexts, common challenges remain. The study proposes a framework to enhance AI chatbot integration in education, noting limitations such as language restrictions and technological expertise.

Durga Prasad Jasti et al. [9] investigate the intersection of conversational AI and cloud platforms, focusing on security and privacy concerns. Conversational AI systems simulate natural discourse through natural language processing (NLP), a facet of artificial intelligence that enables computers to understand and respond to human speech. The manuscript explores how cloud-dependent chatbots are widely integrated into real-world applications, highlighting vulnerabilities and potential threats. It addresses various security challenges associated with cloud environments and chatbot functionalities, detailing specific attacks that pose risks to both systems. This study contributes insights into safeguarding conversational AI systems and cloud platforms amidst evolving technological landscapes.

Rajendran and Chitrarasu [10] delve into the critical role of Natural Language Processing (NLP) in shaping chatbot architecture and functionality. Initially serving as personal virtual assistants, chatbots have become integral to messaging platforms and smart home devices. The chapter explores how NLP drives the development of chatbots by enabling them to interpret user intent, understand contextual cues, and generate appropriate responses in real time. It discusses the technological complexities, challenges, and advancements in NLP algorithms and Natural Language Understanding (NLU) systems. Despite significant progress, the study highlights persistent constraints that limit chatbots' ability to interact and comprehend diverse contexts seamlessly.

3. Objectives

Construct an AI-powered chatbot: Develop a chatbot that can converse with users in natural language and offer tailored support using machine learning algorithms and sophisticated natural language processing techniques.

Combine Several APIs: To improve the chatbot's usefulness and responsiveness, make sure it can connect with many APIs to retrieve and provide real-time data.

Boost User Engagement: To improve accessibility and user engagement, use platforms like WhatsApp that provide simple user contact with the chatbot.

Visualize and comprehend Data: By concentrating on important performance indicators and data patterns, data visualization tools can help comprehend complicated data and enhance user comprehension.

4. Methodology

VehiWalnut's development is centered on a systematic and organized approach that guarantees strength, usability, and effectiveness in the management and prediction of fuel consumption. This section comprehensively examines the methodologies used in the system's development and operation. The Architectural Design of VehiWalnut plays a crucial role in facilitating its functionalities and scalability. The system is designed to be modular, incorporating necessary components and utilizing suitable technologies to enable smooth interaction and data management. The web framework chosen is Flask, primarily for its lightweight nature, flexibility in managing HTTP requests and responses, and its aptness for swift web application development. Due to its simplicity, Flask is an excellent option for implementing VehiWalnut's backend logic. It guarantees efficient communication between the user interface and the underlying system components. SQLite is used as VehiWalnut's database management system. Due to its lightweight and serverless architecture, this technology is highly suitable for applications that prioritize simplicity and easy deployment. SQLite effectively stores and retrieves fuel consumption

data, encompassing fuel type, payment amount, vehicle details, and timestamps for fill-up events. The systematic method of storing data allows for efficient control and retrieval of historical and real-time data, facilitating analysis and reporting tasks.

Twilio API integration allows VehiWalnut to communicate with users through WhatsApp, improving accessibility and user engagement [41]. Twilio offers the essential framework for transmitting and receiving messages, handling user inputs, and delivering prompt responses. This integration not only streamlines user interaction but also expands the scope of VehiWalnut by utilizing a widely adopted messaging platform for data input and reporting [42]. The Python ecosystem encompasses various libraries that play vital roles in various aspects of VehiWalnut. Pandas is a powerful tool for efficiently manipulating and preprocessing data. It ensures that data obtained from user inputs or database queries is properly formatted and structured for further analysis [43].

Matplotlib facilitates the creation of informative visual representations, such as pie charts, histograms, and line plots. These visualizations aid users in comprehending their fuel consumption patterns and trends more easily, enabling them to make informed decisions [44].

Scikit-learn is a software library that implements machine learning algorithms, specifically linear regression models, to predict future fuel requirements using historical consumption data [45]. The strong capabilities of Scikit-learn in training and evaluating models enhance VehiWalnut's predictive analytics functionalities. VehiWalnut's operation relies heavily on efficient data management, which is essential for the precise recording, processing, and storing of fuel consumption data. This includes data from multiple sources, such as user inputs and historical records.

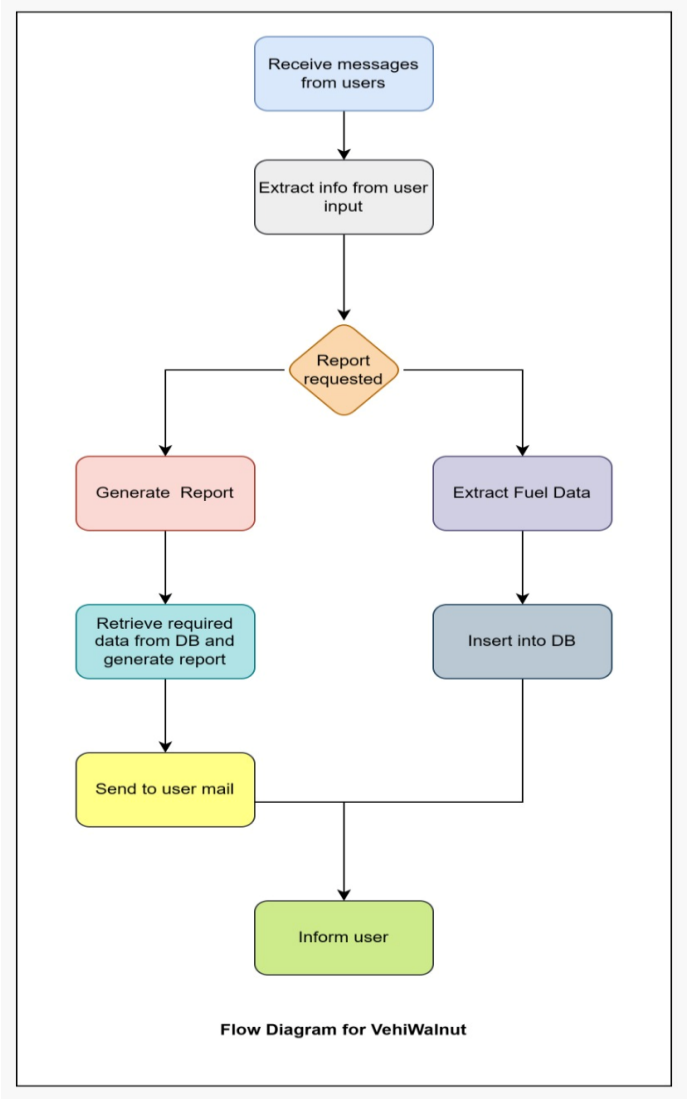

Figure 1: Flow Diagram for VehiWalnut

Data Extraction: VehiWalnut employs regular expressions to extract relevant data from user messages received through WhatsApp. Essential information such as the payment amount, vehicle designation, and distance traveled are identified and extracted to ensure precise processing and storage.

Data validation is performed on the extracted data to verify its accuracy and integrity. Validation checks ensure that the extracted vehicle name matches predefined types and that numerical values, such as payment amounts and kilometers filled, are valid and fall within the expected ranges. The validation process ensures the integrity and dependability of the system's data throughout its operation.

Data Storage: Validated data is stored in an SQLite database, organized with suitable tables and fields to accommodate crucial information regarding fuel consumption. Each entry contains attributes such as fuel type, payment amount, vehicle name, kilometers filled, fill-up date, and time. Timestamps assigned to each entry enable the chronological analysis and retrieval of historical data for reporting and predictive analytics.

The chatbot module in VehiWalnut is vital in improving user engagement and ease of use by incorporating WhatsApp messaging functionalities. VehiWalnut is configured to integrate with the Twilio API to handle incoming and outgoing messages through WhatsApp effectively. Configuration entails establishing a Twilio account, obtaining essential credentials (account SID, auth token, and Twilio phone number), and setting up communication channels to enable smooth message handling and generation of responses.

Message Processing: Incoming messages undergo processing within the Flask application, which functions as the backend server for VehiWalnut. When the system receives a message, it examines the content to ascertain if it pertains to the data input (such as fuel fill-up details) or a request for a specific report (such as fuel consumption analysis). Based on the content of the message, suitable actions are initiated to extract, verify, and store data or create and send reports in response to user inquiries.

Report Generation and Delivery: When users make report requests through WhatsApp, the chatbot component of VehiWalnut retrieves pertinent data from the SQLite database, processes it to produce detailed PDF reports using Matplotlib's PdfPages feature, and sends these reports to users via email. This efficient process guarantees that users obtain comprehensive insights into fuel consumption patterns and trends directly through their preferred communication channel. Implementing machine learning models into VehiWalnut improves its predictive abilities by utilizing historical consumption data to forecast future fuel requirements.

Data Extraction and Preprocessing: The process involves extracting historical fuel consumption data from the SQLite database. This data is then preprocessed to convert date strings into datetime objects. Finally, the data is structured in a format appropriate for machine learning analysis. Feature engineering techniques are utilized to extract supplementary features, such as the time elapsed since the last refueling and the overall distance covered, thereby augmenting the predictive capability of the models.

VehiWalnut utilizes linear regression models from the Scikit-learn library to predict future fuel requirements while considering past consumption patterns. The models are trained using preprocessed data to learn patterns and trends in fuel consumption. This allows them to generate accurate forecasts for upcoming refueling needs. VehiWalnut uses model training to generate forecasts for future refueling dates and consumption trends for each vehicle registered in the system. The generated reports incorporate these insights, offering users actionable information to optimize fuel management strategies and budgeting decisions. Data visualization refers to presenting data in a visual format, such as charts, graphs, and maps, to facilitate understanding and analysis. Report generation, on the other hand, involves the creation of reports that summarize and present data in a structured and organized manner.

VehiWalnut incorporates graphical visualization and comprehensive report generation to present fuel consumption data, enabling users to comprehend and make informed decisions easily. Matplotlib is a powerful tool used for visualizing data. VehiWalnut employs Matplotlib to generate a range of visualizations, such as pie charts, histograms, and line plots. These visual representations aid users in comprehending and analyzing their fuel expenses, refueling durations, and consumption patterns over a period, encouraging the use of data to make informed decisions and develop strategic plans. Generating reports involves utilizing Matplotlib's PdfPages feature to compile comprehensive PDF reports. These reports contain detailed summaries of fuel consumption data and analysis. The report consists of several pages that cover various aspects of fuel usage, such as total spending, refueling durations, predictive analytics, and recent fill-up summaries. The organized structure of these reports guarantees clarity and ease of access to information, enabling users to gain valuable insights for optimizing their vehicle management practices.

Email Delivery: PDF reports are emailed to users using the smtplib module, guaranteeing prompt distribution of insights and recommendations from analyzing fuel consumption patterns. This delivery mechanism improves user convenience and

accessibility by enabling them to examine and respond to comprehensive reports directly from their email inbox. The flow diagram of VehiWalnut outlines the system's overall architecture and workflow. It includes the steps of data extraction, validation, storage, processing, and user interaction through the WhatsApp chatbot (Figure 1).

The development of VehiWalnut follows an organized and systematic approach to integrate technologies, effectively manage data, improve user interaction through chatbot capabilities, utilize predictive analytics for informed decision-making, and ensure reliable and scalable performance. VehiWalnut aims to offer users a powerful platform for efficiently managing and improving fuel consumption. This comprehensive methodology empowers users with valuable insights and enhances their overall experience in vehicle management. The Feature Comparison Table 1 compares VehiWalnut with app-based and traditional fuel management methods, highlighting distinct advantages and functionalities. Table 1 outlines the data input methods, with VehiWalnut using a WhatsApp chatbot, while app-based systems use mobile applications, and traditional methods rely on manual entry.

Database management differences are shown, with VehiWalnut utilizing SQLite for its lightweight, serverless capabilities, compared to the more complex MySQL for app-based systems and Excel for traditional methods. The visualization tools section details VehiWalnut's use of Matplotlib, offering advanced charting options, whereas app-based systems often use Tableau, and traditional methods use basic Excel charts. Table 1 also notes that VehiWalnut includes predictive analytics through linear regression, a feature not typically in app-based or traditional methods. Finally, it highlights user interaction, showcasing VehiWalnut's integration with Twilio API for real-time communication via WhatsApp, compared to the UI-based interaction in apps and email notifications in traditional methods.

5. Algorithm

Step 1: **Setup Environment**

Import Libraries:

Import necessary Python libraries, including Flask, Twilio, SQLite3, datetime, regex, Matplotlib, SMTPLib, Email, NumPy, Pandas, and scikit-learn.

Step 2: **Connect to SQLite Database**

Establish Connection:

Connect to the SQLite database named `fuel_fillups.db`.

Create Tables:

Create the table `fuel_fillups` if it does not already exist using an SQL query.

Step 3: **Setup Twilio for WhatsApp Messaging**

Twilio Credentials:

Define Twilio credentials and initialize the Twilio client.

Step 4: **Process Incoming Messages**

Define `process_message` Function:

Extract and parse relevant information such as amount paid, vehicle name, and kilometers filled from the incoming message using regular expressions.

Handle Report Requests:

Check if the message requests a report, and if so, generate and send the report.

Step 5: **Insert Data into Database**

Define `insert_into_database` Function:

Insert the parsed data (fuel type, amount paid, vehicle name, kilometers filled, fill-up date, fill-up time) into the `fuel_fillups` table.

Step 6: **Generate and Send Report**

Define `generate_report` Function:
Fetch data from the database and generate various graphs and tables, saving them in a PDF file.
Fuel Spending By Vehicle Graph:
Create a pie chart showing fuel spending by vehicle.
Times of Fuel Fillings Graph:
Create a histogram showing the distribution of fill-up times.
Predict Next Fill-up:
Predict the next fill-up date for each vehicle using a Linear Regression model.
Date vs. Amount Paid Graph:
Create a line plot showing the relationship between dates and amounts paid.
Most Recent Fill-ups Table:
Generate a table showing the most recent fill-ups for each vehicle.
Send Report via Email:
Define `send_report_email` function to email the PDF report using SMTP.

Step 7: **Set Up Flask Application**

Define Flask App and Routes:
Define a Flask app with a POST route to handle incoming WhatsApp messages.
Route Logic:
Process incoming messages and send responses using Twilio.

Step 8: **Run the Flask Application**

Run Flask App:
Execute the Flask app to start the server and listen for incoming requests.

The process begins with setting up the environment by importing essential Python libraries, including Flask for the web application framework, Twilio for messaging, SQLite3 for database management, and other data handling and visualization libraries. The SQLite database is connected, and a table named `fuel_fillups` is created if it does not exist. Twilio credentials are defined, and the Twilio client is initialized for WhatsApp messaging. Incoming messages are processed to extract relevant information such as amount paid, vehicle name, and kilometers filled using regular expressions. If a report is requested, the system generates a comprehensive report by fetching data from the database and creating various visualizations and tables, which are saved as a PDF file. This report includes a pie chart of fuel spending by vehicle, a histogram of fill-up times, a prediction of the next fill-up date using a Linear Regression model, a line plot of date versus amount paid, and a table of the most recent fill-ups for each vehicle. The report is then emailed to the user using SMTP. The Flask application has a POST route to handle incoming WhatsApp messages, process the information, and send responses using Twilio. Finally, the Flask app is run to start the server and listen for incoming requests.

6. Experimental setup

6.1. Flask Web Framework

Flask is the primary web framework because of its adaptability and simplicity. Flask is a lightweight Python-based WSGI web application framework that is an excellent choice for developing web applications that require minimal overhead and rapid development. Developers can construct scalable web applications and manage HTTP requests and responses with Flask.

Setup: Flask was installed by executing the command `pip install flask` in the Python package manager (pip).

Configuration: Configuration of the Flask application to manage a variety of routes for a variety of functionalities, including user authentication, data retrieval, and data input. The definition of the configuration settings in a configuration file guarantees a modular and maintainable codebase.

Routing: We established custom routes to handle specific HTTP requests. For instance, we established routes to show the fuel fill-up form, handle form submissions, and produce reports.

6.2. SQLite Database

SQLite is the database management system that stores all pertinent fuel fill-up information. SQLite is a serverless database engine that is self-contained and offers a lightweight solution for data management.

Database Schema: Schema includes tables for storing fuel fill-up records, user information, and vehicle details. The schema included fuel_type, amount_paid, vehicle_name, kilometers_filled, fill_up_date, and time.

Database Initialization: We employed SQLite commands to establish the necessary tables and relationships between them during the database's initialization. We integrated the Flask application with the database to ensure smooth data storage and retrieval.

CRUD Operations: We created functions to allow the database to carry out Create, Read, Update, and Delete (CRUD) operations. These functions enabled the application to insert new records, update existing records, retrieve data for analysis, and delete outdated records.

6.3. Twilio API

Twilio's API is integrated to facilitate user communication via WhatsApp. This integration enables users to input data and interact with the system through WhatsApp messages, improving user engagement and accessibility.

Twilio Installation: To install the Python library for Twilio, execute the command `pip install Twilio`. We established an account on the Twilio platform and acquired the requisite credentials (Account SID and Auth Token).

API Integration: We integrated the Flask application with the Twilio API. We developed functions to streamline the transmission and reception of WhatsApp messages. Because of these functions, the system could accept fuel fill-up data from users and provide real-time responses and confirmations.

Message Handling: We established a webhook to manage incoming messages. The webhook parsed the messages, extracted pertinent data, and initiated the corresponding system actions.

6.4. Data handling and processing

Table 1: Comparative Analysis of Data Processing Methods

Metric	VehiWalnut	App Based	Traditional
Data Extraction Technique	Regex	Keyword Matching	Manual Extraction
Avg. Processing Time (ms)	50	80	100
Error Rate (%)	2	5	10
Flexibility	High	Medium	Low
Scalability	High	Medium	High

Predictive Analysis: The prediction of the next fuel fill-up of all the individual vehicles is made by using the number of days between two consecutive fillings of that particular vehicle, and that number is added to the last fill-up date to predict the approximate next fill-up date. This is done using the linear regression model.

$$Yi=f(Xi,\beta)+ei$$

Where:

Yi = dependent variable
f = function
Xi = independent variable
β = unknown parameters
ei = error terms

Equation 1: Linear Regression (Source: Google)

We implemented machine learning, data manipulation, and visualization using a variety of Python libraries. These libraries comprised Scikit-learn, Matplotlib, and Pandas. Matplotlib is utilized for data visualization. We implemented visualization functions to generate charts and graphs illustrating fuel consumption trends, average fuel costs, and usage patterns over time. We use Scikit-learn for machine learning tasks. We implemented algorithms to classify user behavior, identify outliers, and predict future fuel consumption using historical data.

6.5. Testing and integration

Integration with messaging platforms: The Flask application integrated with popular messaging platforms like WhatsApp, Facebook Messenger, and Slack. This integration allowed users to engage with the system by utilizing their preferred communication channels.

API Connections: We integrated the Flask application with the APIs of each messaging platform. This necessitated the creation of API keys, the configuration of webhooks, and the development of functions to facilitate message sending and receiving.

User Interaction: We developed the system to respond to user inquiries, execute actions based on user inputs, and provide responses. For example, users may request their fuel consumption history, receive maintenance reminders, and receive real-time fuel efficiency tips. Comparative analysis of user feedback (Table 2) and data processing models (Table 3) have been taken and analyzed.

Table 2: Comparative Analysis of User Feedback Mechanisms

Metric	VehiWalnut	App Based	Traditional
User Interaction	WhatsApp	Mobile App	Manual Entry
Feedback Collection Method	Twilio API	In-app Form	Email
Ease of Use	High	Medium	Low

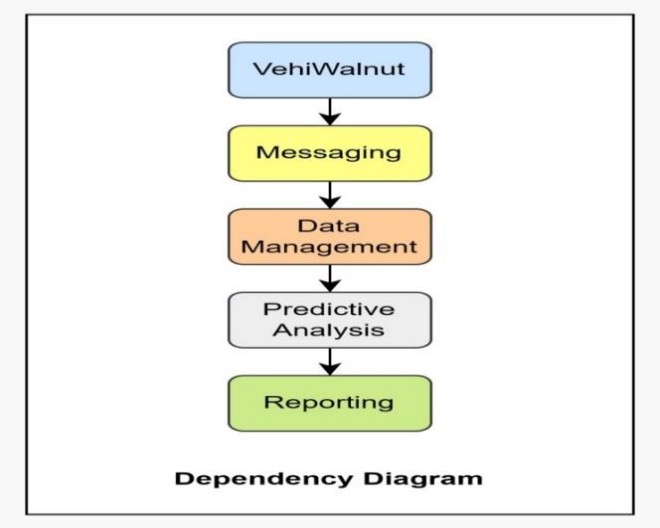

Figure 2: Dependency Diagram for VehiWalnut

The dependency diagram for the VehiWalnut system illustrates the interaction between various components that are integral to its operation. At the user interface level, users communicate through WhatsApp, with messages relayed via the Twilio API to the backend server. The Flask application on the backend processes these messages, extracting and validating data using regular expressions. Validated data is then stored in an SQLite database, the central repository. Matplotlib generates various charts and graphs for data analysis and visualization. These visualizations are compiled into comprehensive PDF reports using Matplotlib's PdfPages module. Finally, the reports are delivered to users via an email service using SMTP. This interconnected

flow ensures seamless data input, processing, visualization, and reporting, enabling efficient fuel management for the users (Figure 2)

7. Results and Discussions

The implementation of the VehiWalnut project yielded significant insights and demonstrated the practical utility of integrating multiple technologies for enhanced vehicle management. The system's architecture, centered on Flask for web development, SQLite for database management, Twilio for WhatsApp integration, and various Python libraries for data analysis and visualization, effectively achieved the project's objectives.

The integration with WhatsApp via Twilio allowed for seamless data collection. Users could input fuel-related information directly through WhatsApp messages, which the system then processed and stored in the SQLite database. This method proved efficient, reducing the need for manual data entry and minimizing errors. The collected data included fuel type, amount paid, vehicle name, kilometers driven, and fill-up dates and times. This comprehensive dataset enabled detailed analysis and visualization.

Table 3: Feature Comparison

Feature	VehiWalnut	App Based	Traditional
Data Input Method	WhatsApp Chatbot	Mobile App	Manual Entry
Database	SQLite	MySQL	Excel
Visualization Tools	Matplotlib	Tableau	Excel
Predictive Analytics	Linear Regression	None	Basic Forecasting
User Interaction	Twilio API	App UI	Email Notifications

Using Matplotlib, we generated various visualizations that provided valuable insights into fuel consumption patterns and expenses. For instance, bar charts displaying the monthly fuel expenses helped users identify periods of high fuel consumption, while line graphs depicting the relationship between kilometers driven and fuel usage highlighted trends that could inform driving habits. The visualizations effectively communicated complex data in an accessible manner, facilitating better decision-making. Implementing machine learning techniques with Scikit-learn, we developed predictive models to forecast future fuel needs. These models analyzed historical data to predict upcoming fill-up dates and potential fuel expenses. Users received notifications about predicted fill-ups, enabling them to plan and manage their fuel budgets more efficiently. The predictive analysis component significantly enhanced the system's value, offering proactive vehicle management capabilities.

The integration of Twilio's API for WhatsApp communication significantly improved user engagement. Users found interacting with the system through a familiar messaging platform convenient, leading to higher data submission rates and more consistent usage. Additionally, the system's ability to send reports and notifications via email ensured that users received timely updates and could review their fuel data at their convenience. SQLite's lightweight and serverless architecture proved beneficial in managing the project's data storage needs. The database efficiently handled the storage and retrieval of large volumes of data, ensuring quick access and reliable performance (Table 4).

Table 4: Risk Analysis and Mitigation Strategies

Risk	Impact	Likelihood	Mitigation Strategy
Data Privacy Breach	High	Medium	Implement end-to-end encryption
System Downtime	Medium	Low	Setup redundant server architecture
Data Loss	High	Low	Regular backups and data recovery plan
User Error	Medium	High	Implement user-friendly interface
API Failure	Medium	Medium	Monitor and have fallback mechanisms

The system's overall reliability was tested through various scenarios, demonstrating its robustness in handling real-time data inputs and generating accurate outputs. Despite its successes, the project faced several challenges. One significant issue was ensuring the accuracy of data entered by users via WhatsApp. Future iterations could implement more sophisticated data validation techniques to mitigate this and offer user-friendly input forms. Additionally, while the predictive models performed

well, their accuracy could be further improved by incorporating additional data points and refining the algorithms. Another area for improvement is the expansion of the system's capabilities. Future versions of VehiWalnut could integrate more advanced analytics, such as anomaly detection for identifying unusual fuel consumption patterns and expand the scope to include maintenance scheduling and cost tracking. User interface design enhancement could also improve the overall user experience, making the system more intuitive and easier to navigate (Figure 3).

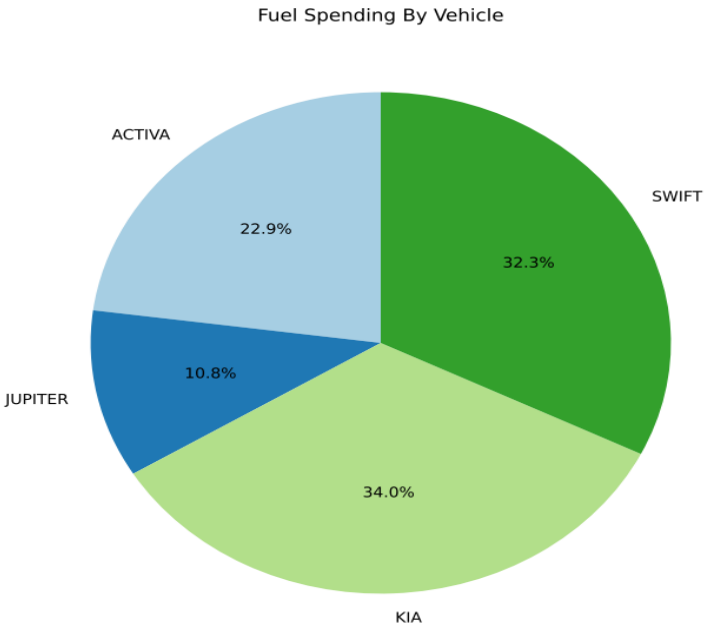

Figure 3: Pie Chart for Fuel Filling Ratio

Several visualizations are generated using Matplotlib, such as pie charts, histograms, and line plots. These charts provide insights into fuel expenses, refueling durations, and consumption patterns over time, aiding users in understanding and optimizing their vehicle management practices (Figure 4).

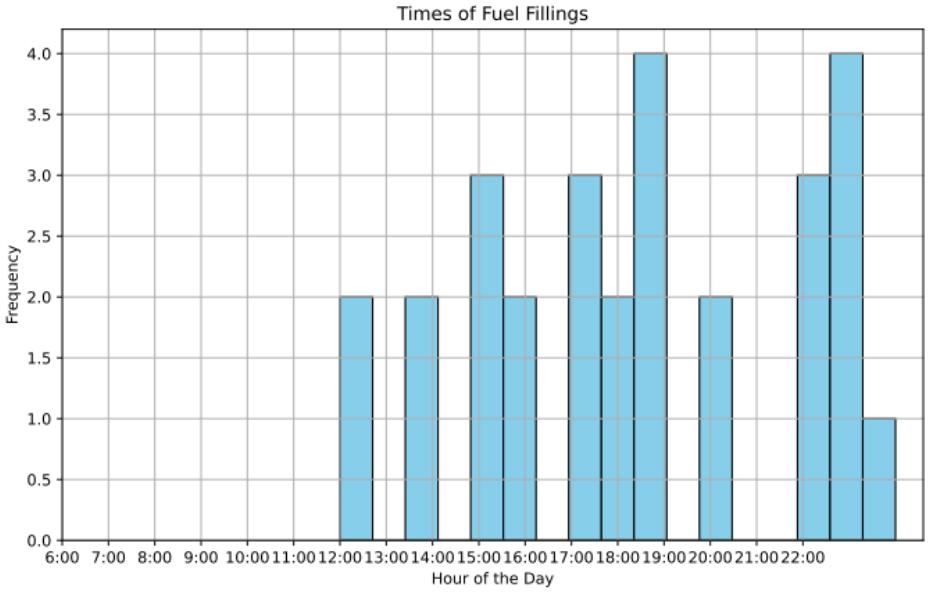

Figure 4: Time of Fuel Fill Ups

“Times of Fuel Fillings” shows the frequency of fuel fill-ups across different hours of the day. The x-axis represents the time of day, ranging from 6:00 AM to 11:00 PM, while the y-axis indicates the number of fuel fillings. The data suggests that fuel fillings are more frequent during the late afternoon and early evening hours, particularly around 4:00 PM and 7:00 PM, with a peak at 7:00 PM. There is also noticeable activity around 3:00 PM and 10:00 PM. The graph provides insights into the typical times when vehicles are refueled, which can help plan fuel stops and manage fuel resources more effectively (Figure 5).

Figure 5: Next, fill up date predictions

The VehiWalnut system provides predictions for the next date when you might need to refuel each of your vehicles. These predictions are based on a Linear Regression analysis of past fuel fill-up data, which calculates the expected number of days between fill-ups (Figure 6).

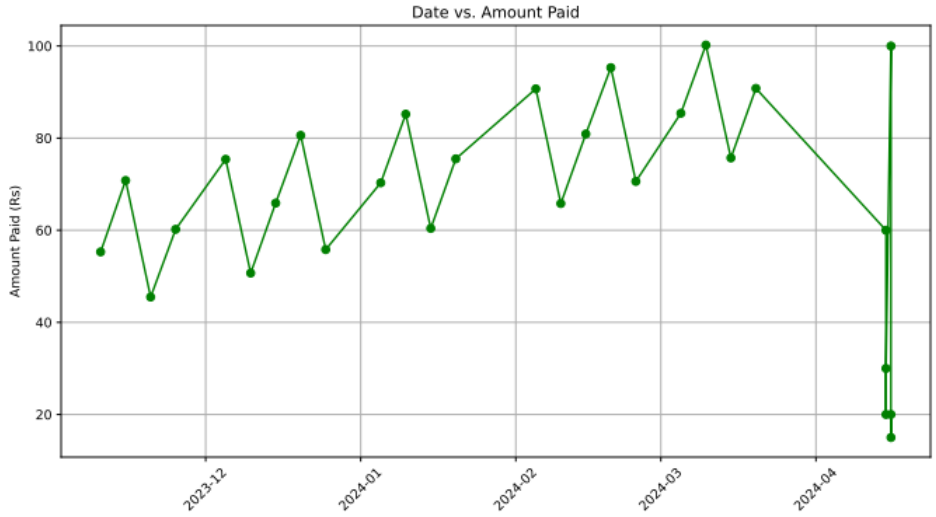

Figure 6: Graph showing Date vs Amount Paid

The system forecasts each vehicle’s next likely refuel date by adding this predicted interval to the most recent fill-up date. This advanced predictive capability helps users proactively manage their fuel needs, ensuring timely refueling and efficient fuel management (Table 5).

Table 5: Most Recent Fill-Up Data

Vehicle Name	Total Amount Paid (RS)	KM When Filling	Fill-up Date	Fill-up Time	Fuel Type
DIESEL	100.2	180	2024-03-10	11:30:00	SWIFT
DIESEL	95.3	170	2024-02-20	14:45:00	SWIFT
DIESEL	90.7	160	2024-02-05	08:30:00	SWIFT
PETROL	100.0	150	2024-04-16	17:35:29	KIA
PETROL	15.0	160	2024-04-16	17:39:57	JUPITER
PETROL	20.0	170	2024-04-16	18:02:57	JUPITER

8. Conclusion

Utilizing cutting-edge technologies, the VehiWalnut project is revolutionizing fuel management. VehiWalnut's fuel monitoring system can easily monitor fuel consumption and uses SQLite, Flask, Python libraries, and Twilio API. It has a cloud-based chatbot that uses machine learning and natural language processing to provide real-time data and support via WhatsApp, improving accessibility and user engagement. By making data entry easier, Twilio's API increases the system's accessibility. Users can comprehend fuel consumption patterns using important technologies such as Pandas, Matplotlib, and Scikit-learn, which facilitate effective data analysis and perceptive visualizations. Machine learning algorithms encourage efficiency and cost savings by predicting future trends and making customized recommendations. The project successfully implemented validation checks and optimization strategies despite obstacles related to data consistency, user adoption, and integration complexity. Upcoming improvements will encompass more sophisticated machine learning models, telematics integration, user feedback customization, and wider platform compatibility. VehiWalnut is an example of how machine learning, databases, and APIs can be combined to produce a useful fuel management system.

Acknowledgment: We extend our heartfelt gratitude to our mentors for their invaluable guidance and support. Special thanks to our dedicated development team for their hard work and creativity. We appreciate our beta testers' crucial feedback, which helped refine VehiWalnut. Lastly, we are grateful to our families and friends for their unwavering encouragement.

Data Availability Statement: The data supporting the findings of this study are available upon request from the corresponding author. Due to privacy and ethical restrictions, certain data may be limited or require additional permissions. Institutional guidelines and data-sharing policies will review and consider all data requests.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding

Conflicts of Interest Statement: The authors declare no conflicts of interest related to this study.

Ethics and Consent Statement: This study was conducted by ethical standards and approved by the relevant institutional review board. Informed consent was obtained from all participants before their involvement in the study.

References

1. P. Gupta and A. Bagchi, "Data Visualization with Python," in Essentials of Python for Artificial Intelligence and Machine Learning, Cham: Springer Nature Switzerland, pp. 237–282, 2024.
2. S. Serbout, F. D. Lauro and C. Pautasso, "Web APIs Structures and Data Models Analysis," 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), Honolulu, HI, USA, pp. 84-91, 2022.
3. G. R. Gunnam, D. Inupakutika, R. Mundlamuri, S. Kaghyan, and D. Akopian, "Assessing performance of cloud-based heterogeneous chatbot systems and A case study," IEEE Access, vol. 12, no.1, pp. 81631–81645, 2024.
4. P. A. Alia, R. W. Febriana, J. S. Prayogo, and R. Kriswibowo, "Implementation chatbot on WhatsApp using artificial intelligence with natural language processing method," ELECTRON Jurnal Ilmiah Teknik Elektro, vol. 5, no. 1, pp. 8–14, 2024.
5. G. Singh, "GCET Virtual Help Desk 'Chat Bot'using Python and Telegram API," Grenze International Journal of Engineering & Technology, vol.3, no.1, p.15, 2024.
6. G. Attigeri, A. Agrawal, and S. V. Kolekar, "Advanced NLP models for technical university information chatbots: Development and comparative analysis," IEEE Access, vol. 99, no.1, pp. 29633–29647, 2024.
7. A. Ali and H. Ali, "Meta-Analysis of Deep Learning Approaches for Machine Learning Chatbots," Easy Chair Prepr, pp. 1–10, 2024.
8. Y. Lin and Z. Yu, "A bibliometric analysis of artificial intelligence chatbots in educational contexts," Interact. Technol. Smart Educ., vol.5, no.1, p-59-78, 2023.
9. V. Durga Prasad Jasti, D. Pounraj, M. Jawarneh, V. H. Meenakshi, P. Prasad, and S. Ray, "Conversational AI and Cloud Platform: An Investigation of Security and Privacy. Conversational Artificial Intelligence," pp. 635–653, 2024.
10. R. K. Rajendran and K. Chitrarasu, "Natural Language Processing (NLP) in Chatbot Design: NLP's Impact on Chatbot Architecture," in Design and Development of Emerging Chatbot Technology, IGI Global, USA, pp. 102–113, 2024.
11. A. G. Usman et al., "Environmental modelling of CO concentration using AI-based approach supported with filters feature extraction: A direct and inverse chemometrics-based simulation," Sustain. Chem. Environ., vol. 2, p. 100011, 2023.
12. A. Gbadamosi et al., "New-generation machine learning models as prediction tools for modeling interfacial tension of hydrogen-brine system," Int. J. Hydrogen Energy, vol. 50, pp. 1326–1337, 2024.

13. B. Prasanth et al., “Maximizing Regenerative Braking Energy Harnessing in Electric Vehicles Using Machine Learning Techniques,” Electronics, vol. 12, no. 5, p.11, 2023.
14. B. Praveen, M. Mohan Reddy Nune, Y. Akshay Kumar, and R. Subash, “Investigating the effect of minimum quantity lubrication on surface finish of EN 47 steel material,” Mater. Today, vol. 38, pp. 3253–3257, 2021.
15. B. S. Alotaibi et al., “Sustainable Green Building Awareness: A Case Study of Kano Integrated with a Representative Comparison of Saudi Arabian Green Construction,” Buildings, vol. 13, no. 9, 2023.
16. D. Ganesh, S. M. S. Naveed, and M. K. Chakravarthi, “Design and implementation of robust controllers for an intelligent incubation Pisciculture system,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 1, no. 1, pp. 101–108, 2016.
17. D. S. Kumar, A. S. Rao, N. M. Kumar, N. Jeebaratnam, M. K. Chakravarthi, and S. B. Latha, “A stochastic process of software fault detection and correction for business operations,” The Journal of High Technology Management Research, vol. 34, no. 2, p.15, 2023.
18. G. S. Sudheer, C. R. Prasad, M. K. Chakravarthi, and B. Bharath, “Vehicle Number Identification and Logging System Using Optical Character Recognition,” International Journal of Control Theory and Applications, vol. 9, no. 14, pp. 267–272, 2015.
19. H. M. Albert et al., “Crystal formation, structural, optical, and dielectric measurements of l-histidine hydrochloride hydrate (LHHCLH) crystals for optoelectronic applications,” J. Mater. Sci.: Mater. Electron., vol. 34, no. 30, p.18, 2023.
20. I. Abdulazeez, S. I. Abba, J. Usman, A. G. Usman, and I. H. Aljundi, “Recovery of Brine Resources Through Crown-Passivated Graphene, Silicene, and Boron Nitride Nanosheets Based on Machine-Learning Structural Predictions,” ACS Appl. Nano Mater., 2023.
21. J. I. D. Raj, R. B. Durairaj, S. V. Ananth, and P. Barmavatu, “Experimental investigation of the effect of e-waste fillers on the mechanical properties of Kenaf woven fiber composites,” Environ. Qual. Manage., vol. 34, no. 1, p.22, 2024.
22. J. Immanuel Durai Raj, R. I. B. Durairaj, A. John Rajan, and P. Barmavatu, “Effect of e-waste nanofillers on the mechanical, thermal, and wear properties of epoxy-blend sisal woven fiber-reinforced composites,” Green Process. Synth., vol. 12, no. 1, p.14, 2023.
23. J. Usman, S. I. Abba, N. Baig, N. Abu-Zahra, S. W. Hasan, and I. H. Aljundi, “Design and Machine Learning Prediction of In Situ Grown PDA-Stabilized MOF (UiO-66-NH2) Membrane for Low-Pressure Separation of Emulsified Oily Wastewater,” ACS Appl. Mater. Interfaces, Mar. 2024.
24. K. Subramanian, N. Meenakshisundaram, and P. Barmavatu, “Experimental and theoretical investigation to optimize the performance of solar still,” Desalination Water Treat., vol. 318, no. 100343, p. 100343, 2024.
25. K. Subramanian, N. Meenakshisundaram, P. Barmavatu, and B. Govindarajan, “Experimental investigation on the effect of nano-enhanced phase change materials on the thermal performance of single slope solar still,” Desalination Water Treat., vol. 319, no. 100416, p. 100416, 2024.
26. M. A. Tripathi, K. Madhavi, V. S. P. Kandi, V. K. Nassa, B. Mallik, and M. K. Chakravarthi, “Machine learning models for evaluating the benefits of business intelligence systems,” The Journal of High Technology Management Research, vol. 34, no. 2, p.14, 2023.
27. M. A. Yassin et al., “Advancing SDGs : Predicting Future Shifts in Saudi Arabia’s Terrestrial Water Storage Using Multi-Step-Ahead Machine Learning Based on GRACE Data,” 2024.
28. M. A. Yassin, A. G. Usman, S. I. Abba, D. U. Ozsahin, and I. H. Aljundi, “Intelligent learning algorithms integrated with feature engineering for sustainable groundwater salinization modelling: Eastern Province of Saudi Arabia,” Results Eng., vol. 20, p. 101434, 2023.
29. M. Akbar, I. Ahmad, M. Mirza, M. Ali, and P. Barmavatu, “Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach,” Cluster Comput., vol. 27, no. 3, pp. 3683–3702, 2024.
30. M. Akbar, M. M. Waseem, S. H. Mehanoor, and P. Barmavatu, “Blockchain-based cyber-security trust model with multi-risk protection scheme for secure data transmission in cloud computing,” Cluster Comput., 2024, Press.
31. M. Chakravarthi and N. Venkatesan, “Design and implementation of adaptive model based gain scheduled controller for a real time non linear system in LabVIEW,” Research Journal of Applied Sciences, Engineering, and Technology, vol. 10, no. 2, pp. 188–196, 2015.
32. M. Chakravarthi and N. Venkatesan, “Design and Implementation of Lab View Based Optimally Tuned PI Controller for A Real Time Non Linear Process,” Asian Journal of Scientific Research, vol. 8, no. 1, p.11, 2015.
33. N. Sirisha, M. Gopikrishna, P. Ramadevi, R. Bokka, K. V. B. Ganesh, and M. K. Chakravarthi, “IoT-based data quality and data preprocessing of multinational corporations,” The Journal of High Technology Management Research, vol. 34, no. 2, p.15, 2023.
34. N. Venkatesan and M. K. Chakravarthi, “Adaptive type-2 fuzzy controller for nonlinear delay dominant MIMO systems: an experimental paradigm in LabVIEW,” Int. J. Adv. Intell. Paradig., vol. 10, no. 4, p. 354, 2018.
35. P. Rex, M. K. Rahiman, P. Barmavatu, S. B. Aryasomayajula Venkata Satya Lakshmi, and N. Meenakshisundaram, “Catalytic pyrolysis of polypropylene and polyethylene terephthalate waste using graphene oxide-sulfonated zirconia

(GO-Szr) and analysis of its oil properties for Bharat Stage VI fuel production," Environ. Qual. Manage., vol. 33, no. 4, pp. 501–511, 2024.
36. P. Rex, N. Meenakshisundaram, and P. Barmavatu, "Sustainable valorisation of kitchen waste through greenhouse solar drying and microwave pyrolysis– technology readiness level for the production of biochar," J. Environ. Health Sci. Eng., 2024, Press.
37. R. Jain et al., "Internet of Things-based smart vehicles design of bio-inspired algorithms using artificial intelligence charging system," Nonlinear Eng., vol. 11, no. 1, pp. 582–589, 2022.
38. S. Buragadda, K. S. Rani, S. V. Vasantha, and K. Chakravarthi, "HCUGAN: Hybrid cyclic UNET GAN for generating augmented synthetic images of chest X-ray images for multi classification of lung diseases," Int. J. Eng. Trends Technol., vol. 70, no. 2, pp. 229–238, 2022.
39. S. Das, R. K. Ghadai, G. Sapkota, S. Guha, P. Barmavatu, and K. R. Kumar, "Optimization of CNC turning parameters of copper–nickel (Cu–Ni) alloy using VIKOR, MOORA, and GRA techniques," Int. J. Interact. Des. Manuf. (IJIDeM), 2024, Press.
40. S. I. Abba et al., "Integrated Modeling of Hybrid Nanofiltration/Reverse Osmosis Desalination Plant Using Deep Learning-Based Crow Search Optimization Algorithm," Water (Switzerland), vol. 15, no. 19, p.11, 2023.
41. S. I. Abba, A. G. Usman, and S. IŞIK, "Simulation for response surface in the HPLC optimization method development using artificial intelligence models: A data-driven approach," Chemom. Intell. Lab. Syst., vol. 201, no. April, 2020.
42. S. I. Abba, J. Usman, and I. Abdulazeez, "Enhancing Li + recovery in brine mining : integrating next-gen emotional AI and explainable ML to predict adsorption energy in crown ether-based hierarchical nanomaterials," pp. 15129–15142, 2024.
43. S. Ohol, V. K. Mathew, V. Bhojwani, N. G. Patil, and P. Barmavatu, "Effect of PCM-filled hallow fin heat sink for cooling of electronic components — a numerical approach for thermal management perspective," Int. J. Mod. Phys. C., 2024, Press.
44. T. Prasad, B. Praveen, Y. A. Kumar, and K. Krishna, "Development of carbon and glass fiber-reinforced composites with the addition of nano-egg-shell powder," in Lecture Notes in Mechanical Engineering, Singapore: Springer Nature Singapore, pp. 569–577, 2022.
45. U. B. Vishwanatha, Y. D. Reddy, P. Barmavatu, and B. S. Goud, "Insights into stretching ratio and velocity slip on MHD rotating flow of Maxwell nanofluid over a stretching sheet: Semi-analytical technique OHAM," J. Indian Chem. Soc., vol. 100, no. 3, p. 100937, 2023.