Enhancing Mobile Efficiency: A Cloud-Powered Paradigm for Extended Battery Life and Enhanced Processing Capabilities

Dua Agha[1], Veena Kumari[1], Areej Fatemah Meghji[1*]

[1]Department of Software Engineering, Mehran University of Engineering and Technology Jamshoro, Pakistan

* **Correspondence**: Areej Fatemah Meghji, areej.fatemah@faculty.muet.edu.pk

Citation | Agha. D, Kumari. V, Meghji. A. F, "Enhancing Mobile Efficiency: A Cloud-Powered Paradigm for Extended Battery Life and Enhanced Processing Capabilities", IJIST, Vol. 6 Issue. 1 pp 58-69, Feb 2024

Received | Jan 10, 2024 **Revised |** Feb 03, 2024 **Accepted |** Feb 05, 2024 **Published |** Feb 09, 2024.

In an interconnected world where mobile phones are essential to everyday operations, the constraints of these devices in terms of processing power, memory, storage, and energy efficiency are becoming increasingly apparent. This research introduces an innovative solution by integrating Mobile Cloud Computing (MCC) to address these challenges. The research focuses on the creation of an Android application called "ServiVerse" that efficiently drains the phone's battery to imitate real-world conditions. The software is accompanied by a Firebase-connected battery optimizer, which provides users with complete insights into battery state, cleaning history, and graphical representations of performance. The system's distinguishing feature is outsourcing power-intensive operations to a cloud server, resulting in increased energy efficiency and battery life. The study demonstrated successful battery optimization tactics adapted to individual users, such as the amount of cache and RAM deleted and storage space freed up on the mobile devices. This strategy has proven to be vital in addressing a key concern about background processing and the loss of power generation on mobiles, which is providing users with more efficient and longer-lasting battery life.

Keywords: Cloud Computing, Mobile Cloud Computing, Battery Optimization, Offloading, Firebase.

Author's Contribution
Dua Agha: Writing – original draft, methodology, experimentation, visualization, literature review, result reporting. Veena Kumari: Writing original draft, methodology, experimentation, visualization, literature review, and result reporting. Areej Fatemah Meghji: Methodology, writing – editing and review, validation, result reporting.

Acknowledgement: Nil

Conflict of Interest: The authors declare they have no conflict of interest in publishing this manuscript in IJIST.

Project details: This research was not part of any project.

Introduction:

In today's dynamic world, mobile phones serve as essential tools for various daily tasks, fostering communication, interactions, and modern learning experiences. The rise of Mobile Cloud Computing (MCC), driven by widespread mobile service adoption and cloud integration, addresses inherent limitations in mobile devices. These limitations include computational power, memory, storage, and energy efficiency, requiring a nuanced approach for enhanced functionality [1]. The synergy of mobile and cloud computing proves pivotal in overcoming challenges, especially in the context of battery technologies lagging behind innovation [2][3]. To improve smartphone battery efficiency, designers integrate hardware- and software-based energy optimization approaches, evaluating components like Bluetooth, CPU, and camera consumption [4]. The integration of cloud computing technology emerges as a solution to tackle these challenges, gaining traction in research due to the widespread use of mobile phones [5][6].

Corporate productivity has significantly increased with the adoption of internet-based cloud server infrastructures, meeting the demand for efficient data management and accessibility. Cloud-based systems, offering virtual services, reduce dependence on physical products and hardware, contributing to enhanced energy efficiency and reduced emissions [7]. Multiple tasks can be carried out concurrently through cloud computing's mobility, saving CPU, storage, and battery power on the device. Computational offloading in mobile devices streamlines background processing, enhancing server-based system performance and resource utilization.

The rapid evolution of computational functionalities in mobile phones has raised user expectations, leading to increased battery drainage and performance issues [8]. The decline in processing power exacerbates battery life limitations, disrupting daily routines. Addressing this challenge is imperative, necessitating a focus on lowering energy consumption at the software level. This research undertakes the task of optimizing mobile phone power usage through cloud computing, facilitating automatic adjustments without requiring manual intervention. A user-friendly Android application is developed that analyzes and curtails battery-consuming processes, including cache and background operations of intensive applications. The application offers analytical insights and graphical representations of battery status, along with a list of battery-consuming applications, helping users to identify potential drains and providing options to halt unnecessary processes. This research seeks to bridge the gap between mobile technology and efficient battery management, offering users a reliable solution to enhance their mobile phone experience by creating an intuitive application that addresses battery consumption effectively.

The paper is organized into the following sections: we start by presenting a "Literature Review" which offers a comprehensive overview of pertinent literature to establish the context for the current research. This is followed by the specific goals of the research outlined in "Objectives", followed by "Material & Methods" which elaborates on the procedures and steps followed during the research. The outcomes of the study are detailed in "Results and Discussion" while "Conclusion" presents a summary of the research and also addresses the Limitations and Future Directions providing insights into potential areas for further exploration.

Literature Review:

To improve mobile battery efficiency and processing power, several techniques and methods have been carried out to address the issue. One of the used technologies is CloneCloud technology, which presents an adaptable application partitioner and execution runtime that allows unaltered applications to smoothly offload processing from mobile devices to computing cloud-connected device clones. Clone Cloud optimizes and partitions programs using static analysis and dynamic profiling, enabling efficient cloud integration. According to the analysis, Clone Cloud may speed up smartphone applications by up to 21.2x and support various input and network partitions [9]. Another study provides solutions with different aspects such as

Mobile Edge Computing that provides remote computing capacity close to smart mobile devices (SMDs). In this study, a single-cell Mobile Edge Computing system that supports multiple SMDs is examined. Each SMD has a distinct task that can be delegated to a MEC server for processing. In comparison to local execution and complete task offloading, the heuristic scheme known as Overall Energy Minimization by Resources Partitioning (OEMRP) is evaluated. The proposed solution also minimizes rejected SMDs with insufficient radio resources, saving approximately 48% of energy compared to local execution and 24% and 26% of energy compared to offloading all tasks [10].

To increase battery life for mobile computing, the paper presents an energy-optimal offloading technique that runs applications on remote servers. The study gives a mathematical model for the issue and explores the use of HetNets for quick and easy wireless access. In simulations, the optimal algorithm, which is based on combinatorial optimization, saves 43% of energy. The O(N) complexity suboptimal algorithm comes close to the ideal solution. Additionally, the article explains how energy conservation and SINR in LTE networks are related, showing how the suggested algorithms can perform better under optimal wireless conditions [11].

To improve response time and energy management for mobile cloud computing, the paper proposed a framework that prompts for segregating mobile devices according to fuzzy K-nearest neighbors and enhancing computational offloading with the Hidden Markov Model and Ant colony optimization. When compared to existing techniques, the algorithm performs 89% better in terms of response time, 95% better in terms of energy consumption, and 50% better in terms of processing cost. This method shows the shortest route between resources and mobile devices [12].

The Mobile Capabilities Augmentation using Cloud Computing (MCACC) framework, which divides mobile applications into services for local or remote execution has been introduced in [13]. The model considers real-time metrics like execution time, consumption of energy, battery lasting, memory, and security. The results demonstrate that both heavy and light applications can benefit from the MCACC model, saving energy and improving performance. Network bandwidth affects offloading, with high bandwidth resulting in better offloading. The experimental results show that executing the requested service on the mobile, especially with low network bandwidth, saves mobile processing power [13]. In the area of Mobile Edge Computing (MEC), a potential optimization model is proposed that considers task dependency and dynamic energy harvesting using wireless power transfer to minimize task completion time and includes a greedy algorithm for offloading subtasks with dependencies to the location with the quickest turnaround time. Using a custom simulated annealing algorithm, the algorithm outperforms task completion time, and algorithm running time drops by at least 9.32% on different task graphs and by at least 11.47% after further optimization by the simulated annealing algorithm [14].

Another research introduces a MEC architecture that combines computation offloading and artificial intelligence technology. The LSTM algorithm performs task prediction based on node performance and data size of tasks from mobile users. It also strategies migrations of tasks for edge cloud scheduling. The architecture controls total task delay with growing data and subtasks, ensuring the completion of prior tasks [15]. To obtain task optimization, a novel task-scheduling algorithm is utilized with its dynamic decision-based methods to reduce energy consumption and time execution along with a task scheduling server to offload computation to the cloud, improving performance and resource utilization on mobile devices. Jobs beyond time limits are moved to cloud VMs, saving mobile devices' limited battery power [16]. The competence of mobile computing hybrid applications is evaluated at runtime in this research's technique to address the issue of task-intensive applications allowing arbitrary configurations to optimize the efficiency trade-off. The experimental analysis takes into consideration two MC hybrid applications that are both executed using a straightforward MC framework and are modularized based on computationally demanding tasks. Results conclude that effective configurations are manageable. Using a middleware-based task delegation from mobile to cloud, the technique involves parsing down the source code of an application into smaller units based on tasks that use battery power [17]. For scheduling tasks in a collaborative environment, a ranking algorithm is proposed to evaluate peer devices' capabilities and send jobs to the most appropriate devices using a tweaked Hungarian algorithm. For resource-intensive applications, a local cloud architecture with scheduling plans is utilized to distribute workload and optimize resource consumption. This collaborative environment enables all devices to work together to overcome their limits and enable effective local operations without the need for external assistance from cloud services [18].

The research by Kaur et al. concludes that to make your mobile phone more efficient and to improve its processing capabilities cloud computing along with the data synchronization between the mobile and cloud can be helpful. The data from a mobile phone can be automatically synchronized to the user's devices when they are connected to the internet along with consuming minimum energy usage. Offloading meaningful data can be partial or complete over the cloud reduces the workload of smartphone applications and it results in less battery consumption with extended battery life along with more reliability [19].

Objectives:

The objective of this study is to investigate the integration of MCC as a solution to mitigate mobile device limitations, specifically addressing issues related to battery drainage and performance. To this end, the research aims to develop a custom power-intensive Android application designed to efficiently drain a phone's battery, serving as a simulation tool. Additionally, the research also aims to create a Battery Optimizer application connected to Firebase to offer users insights into mobile battery life and consumption. The goal of the Optimizer is to intelligently identify power-hungry components of the mobile phone, including both the custom power-intensive app and mobile internal/external applications. The goal of the optimizer is to offload the power-hungry components of the mobile phone to a cloud server through the fusion of cloud computing. By executing resource-intensive applications on the cloud server rather than the device, the aim is to significantly save power, extend battery life, and enhance processing capabilities, thereby providing users with more effective and longer-lasting battery performance.

Material and Method:

The current research aims to significantly save power, extend battery life, and enhance processing capabilities, providing users with more effective and longer-lasting battery performance. Figure 1 depicts the workflow diagram of the proposed system, outlining the sequence of operations. It includes a customized, power-intensive application that integrates Bluetooth, Wi-Fi, location services, sensors, and timers. Moreover, it identifies internal and external applications running on the phone, all interconnected with a battery optimizer. The battery optimizer is further linked to the cloud (Firebase), serving as a repository for cleanup information specific to each user. Our research incorporates two main domains into the development of the proposed model.

Mobile Cloud Computing:

Mobile Cloud Computing is designed to enable powerful mobile applications or services to run on various electronic devices. Through the use of cloud platforms and the internet, this technology expands data processing and storage capabilities beyond mobile devices [20]. Typically, mobile-cloud hybrid apps are managed by third parties in remote data centers, handling data storage and computational responsibilities. This strategy guarantees data integration, accessibility, security, and uptime [21]. These applications can operate over the Internet but require regular updates. Mobile cloud computing offers a desktop-like experience on mobile devices and tablets while maintaining portability. It enhances efficiency, reduces costs for businesses by connecting to more network providers, and adds additional features to mobile phones and other electronic devices.

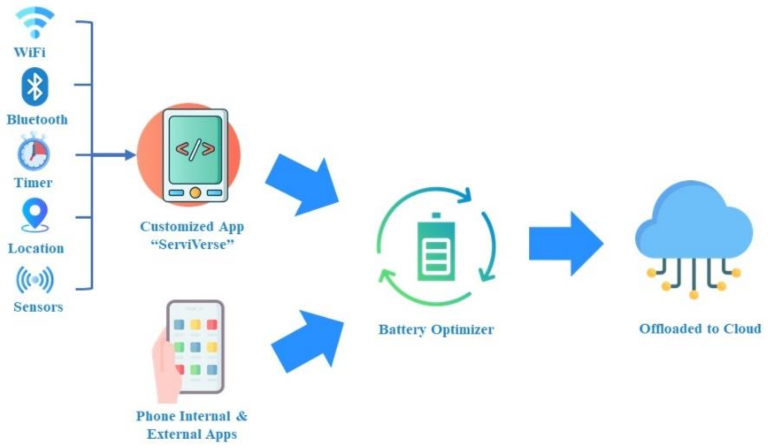


Figure 1: System Workflow Diagram

Firebase:

Firebase is a platform designed for the creation of mobile and web applications, offering convenient access to data storage. It functions as a real-time database that is accessible through the internet and offers effective data and binary file storage. Google's mobile platform, Firebase, helps people and companies create high-quality applications and grow their operations [22]. Data is handled as streams in Firebase, while Google Cloud Storage is used to manage file storage. The process of storing the cache cleaning session in Firebase is executed in an encrypted format, ensuring heightened user security when data is stored in the cloud.

Results and Discussion:

Our system comprises two key components: a custom power-intensive application named "SERVIVERSE", and a battery optimizer connected to Firebase for data storage.

Custom Power-Intensive Application:

ServiVerse, designed to drain the phone's battery efficiently, utilizes built-in Android services such as Bluetooth, Wi-Fi, location services, sensors, and timers. Essential components of the application interface include a splash screen that displays a graphical representation when it launches, a home screen with a user-friendly dashboard made with grid layout and card views, and specialized functionality for Bluetooth, Wi-Fi, location, sensors, and timers depicted in Figure 2.

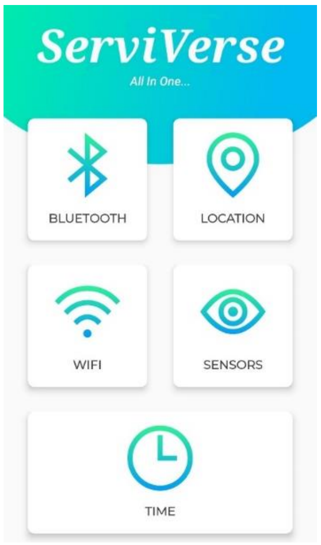


Figure 2: ServiVerse Features

Users may communicate with associated devices, control Wi-Fi connections, obtain textual map data with satellite views, retrieve different sensor readings, and use stopwatches and alarms by using these functions. Our unique, resource-intensive program "ServiVerse" has fundamental functions to provide users with a full user experience. To accurately achieve battery depletion, it periodically modifies its power consumption patterns to simulate practical situations when various processes run simultaneously in a mobile phone.

Use of Bluetooth feature to link devices for file transfer, the Wi-Fi module to mimic data-intensive actions like downloading huge files or streaming high-definition media. Furthermore, the app simulates ongoing GPS monitoring, giving important details about location-based power usage. The sensors show the impacts of sensor-intensive applications on battery life in addition to reading proximity, acceleration, temperature, humidity, brightness, pressure, magnetic effects, and RGB data. Additionally, the timer function not only includes a stopwatch but also integrates various alarms with diverse pre-installed ringtones, showcasing the power usage implications of timer-based applications.

Battery Optimizer:

Although there are already mobile phone battery optimizers available that improve performance by clearing cache, none of them include a built-in feature associated with the cloud. Our optimizer has an integrated function linked to the cloud (Firebase), where all the cached data is securely stored in encrypted format once it is offloaded. Figure 3 illustrates the phases involved in the battery optimizer development process using Android Studio.

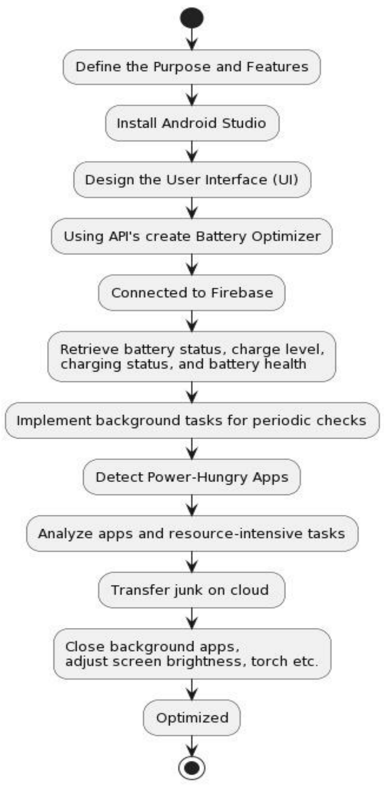
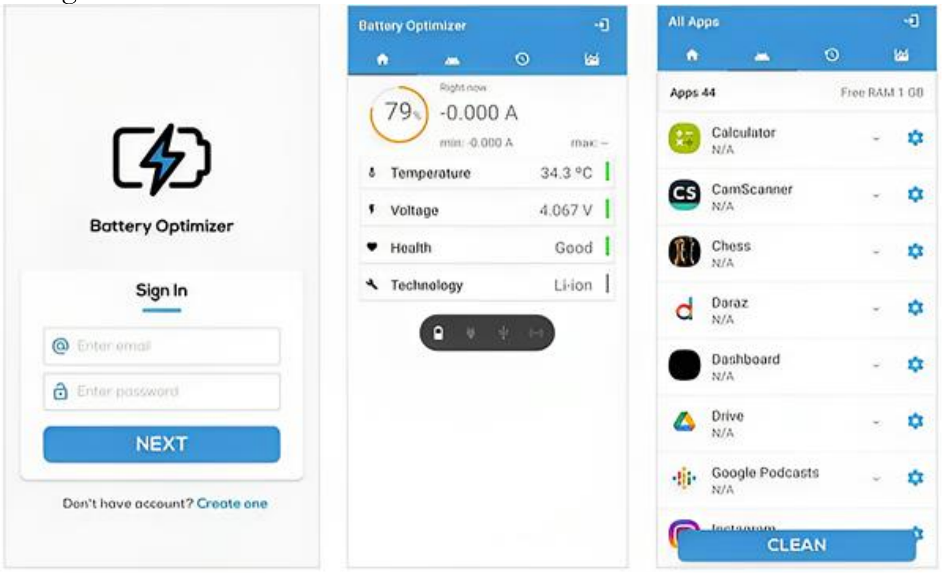

Define the Purpose and Features
Install Android Studio
Design the User Interface (UI)
Using API's create Battery Optimizer
Connected to Firebase
Retrieve battery status, charge level, charging status, and battery health
Implement background tasks for periodic checks
Detect Power-Hungry Apps
Analyze apps and resource-intensive tasks
Transfer junk on cloud
Close background apps, adjust screen brightness, torch etc.
Optimized

Figure 3: Battery Optimizer Development Procedure

In contrast, the battery optimizer, which is integrated with Firebase, comprises numerous interactive screens. The splash screen serves as an introductory interface during application loading, followed by a signup screen where users can register their personal information, enabling seamless integration with the system. With the use of a username and password, the login page guarantees safe user authentication. The optimizer offers comprehensive information on the phone's charging state, including battery level, temperature, voltage, health, and technology. Additionally, it also provides a list of internal and external programs, giving users the option to authorize access or change settings for each one individually as shown in Figure 4.



Right now
79%
-0.000 A
min: -0.000 A max: –

Temperature	34.3 °C
Voltage	4.067 V
Health	Good
Technology	Li-ion

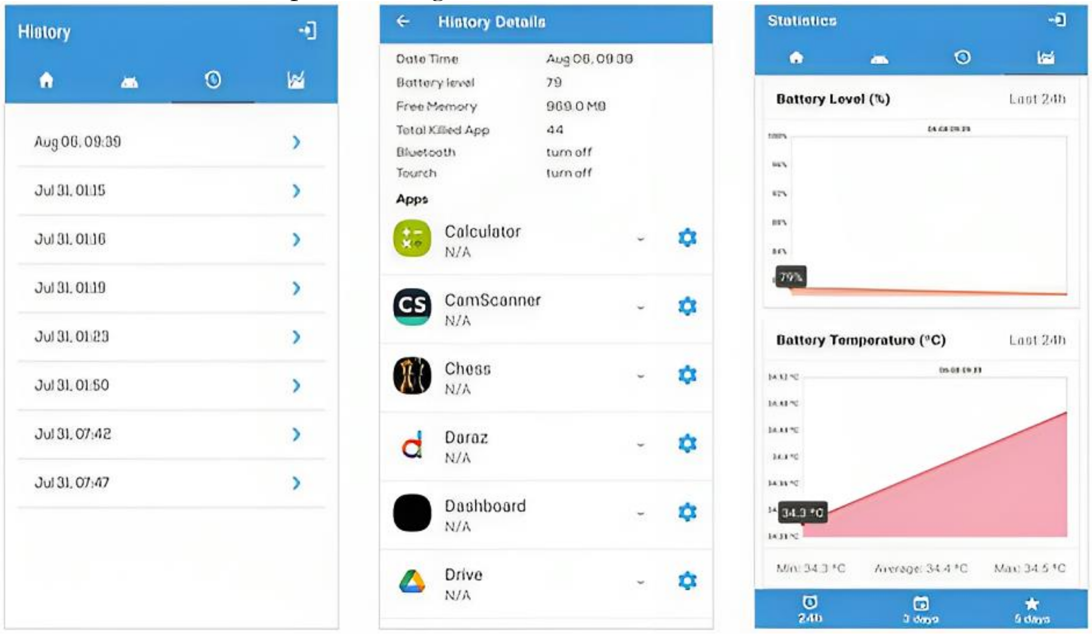
Apps 44 Free RAM 1 GB

Calculator	N/A
CamScanner	N/A
Chess	N/A
Daraz	N/A
Dashboard	N/A
Drive	N/A
Google Podcasts	N/A
Instagram	

Figure 4: Battery Optimizer-Fetching

The program data is saved chronologically, displaying the history of cleaning, together with cache and RAM details. Although it was initially intended to transfer program components to the cloud, Android constraints have limited what can be saved there to just useful information. This includes information like the status of Bluetooth and torches, as well as wiping history, clearing space, and background applications that have been stopped.

Moreover, to help users keep track of their device's performance, the optimizer also provides graphical representations of battery status and temperature over the last 24 hours, 3 days, or 5 days. All of these details, such as cleaning history, cache removal, space freed up, and the status of background programs, are meticulously logged and connected to specific user accounts on Firebase as depicted in Figure 5.



Aug 06, 09:39
Jul 31, 01:15
Jul 31, 01:16
Jul 31, 01:19
Jul 31, 01:23
Jul 31, 01:50
Jul 31, 07:42
Jul 31, 07:47

Date Time	Aug 06, 09:39
Battery level	79
Free Memory	969.0 MB
Total Killed App	44
Bluetooth	turn off
Tourch	turn off

Apps

Calculator	N/A
CamScanner	N/A
Chess	N/A
Daraz	N/A
Dashboard	N/A
Drive	N/A

Min: 34.3 °C Average: 34.4 °C Max: 34.5 °C

Figure 5: Battery Optimizer-History and Graphical Representation

When employing a battery optimizer to enhance mobile performance, various key pieces of information are extracted. Firstly, the cleanup timestamps are logged to record the optimization process. Additionally, the battery level after each cleanup is depicted, presenting insights into the device's power status during this operation. The optimizer also reveals the amount of memory that has been liberated, indicating the efficiency of the cleanup in optimizing storage. It identifies background applications that were consuming battery resources, specifying the total number of terminated apps that impact on overall battery. Notably, features like Bluetooth or the torch are automatically turned off, contributing to a more comprehensive and effective optimization approach. This makes certain efficient battery management and provides users with valuable insights into their device's power consumption patterns.

Overall, our research implementation not only focuses on draining the battery for analysis but also delves deeper into real-world usage scenarios. Through interactive notifications and detailed graphical representations, our system ensures active user participation, ultimately leading to more effective battery optimization strategies tailored to individual user's needs and habits.

Illustrated in Figure 6 is a thorough overview of the hierarchical organization within the Battery Optimizer Backend, delineating the interconnections and structure among its various components. Throughout the development process in Android Studio, the Main Activity class of the battery optimizer assumes a pivotal role, with the 'DataEstimator' class estimating device and battery-related data through the mApp object. The 'NetworkWatcher' class manages server status, device registration, and data transmission to the Firebase server via the 'CommunicationManager' class. The 'OnRefreshComplete' method updates the UI after completing a task list refresh, and 'loadComponents' initialize necessary components and data for the activity, such as setting up the database, services, permissions, and event listeners.

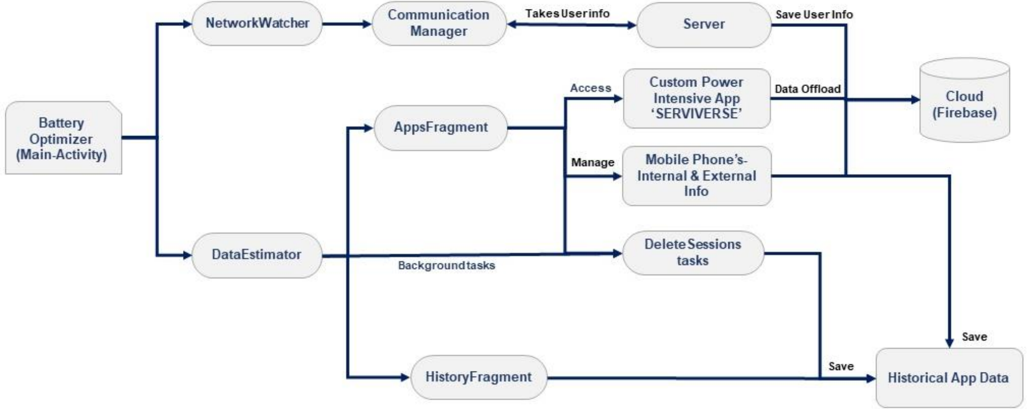


Figure 6: Battery Optimizer-Backend Hierarchy

The 'AppsFragment' handles the power-intensive app "ServiVerse" along with managing tasks related to phone internal and external information, such as running processes, killing apps, and handling permissions for device resources using the 'getMemory' method. It includes a cleanup process triggered by a clean button click, which disables Wi-Fi and Bluetooth, kills apps, logs device information, and performs other operations like turning off the camera torch and refreshing app data. The 'DeleteSessionsTask' class executes the deletion of outdated battery sessions in the background, and the 'HistoryFragment' displays historical app data, allowing users to view additional information by clicking on items.

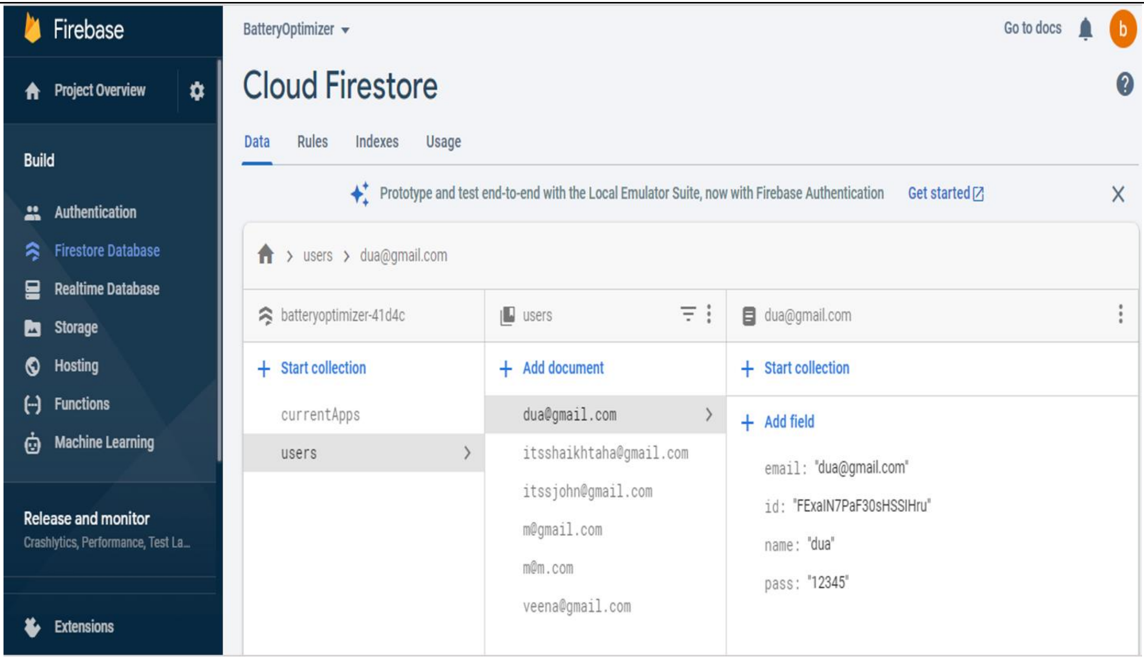
The 'CommunicationManager' class orchestrates the uploading of data samples to the firebase, managing communication flow and providing status updates to the UI. The 'RegisterDeviceHandler' class is responsible for device registration, making API calls using Retrofit, and gathering device-specific information for the registration payload, including model, manufacturer, OS version, and root status. Overall, these classes work together to ensure effective communication with Firebase, data estimation, and management of app-related tasks on the Android device.

Discussion:

This system's standout feature is its ability to offload power-intensive mobile tasks to Firebase, a cloud server. Firebase meticulously records and organizes information, tracking each user's phone cleaning activities. Figure 7 and Figure 8 visually represent the frequency of cleaning sessions, the amount of cache and RAM eliminated, and the storage space freed up. The process of storing the cache cleaning session in Firebase is executed in an encrypted format, ensuring heightened user security when data is stored in the cloud. This not only enhances energy efficiency but also extends battery life. The approach is sustainable, as it reduces user's device strain, ensures secure data storage, and maintains user privacy. The cloud infrastructure incorporates a default safety mechanism to enhance security for its users. This implies that our optimizer, seamlessly integrated with the cloud, ensures security for users. Each user undergoes a distinct registration process within the optimizer, ensuring personalized and separate user profiles. Once each user has performed their respective cleansing operations, the data is safely offloaded to the cloud in an encrypted format, with each user's information stored separately.

The tracked user activity provides valuable insights for continuous optimization and the visual feedback fosters transparency. The integration with Firebase streamlines data management, ensuring immediate updates and accessibility. As the system promotes responsible and efficient resource utilization, it contributes to better performance. The steady monitoring of cleaning activities establishes a robust feedback loop for further developments. The system also includes a feature that automatically turns off the Torch and Bluetooth connections within the device, enhancing both security and energy efficiency. This comprehensive approach not only ensures user's devices are optimized for performance but also showcases valuable insights into user behavior and preferences, aiding in continuous improvement and customization of the optimization process.

This study holds significant implications across various domains. Firstly, it focuses on mobile device optimization, which aims to improve mobile device performance and battery life. The influence extends to user experience improvement, where graphical representations and insights into energy utilization allow users to make educated decisions regarding their device's performance. Furthermore, the study promotes energy efficiency by integrating cloud computing for power-intensive tasks. The usage of Firebase demonstrates a priority on data security and privacy, with default safety measures and individualized user profiles offering comprehensive data protection for users. Overall, this study proposes a comprehensive approach that seamlessly integrates cloud computing, data analytics, and user-friendly applications to overcome the limits of mobile devices while improving overall performance and battery life.

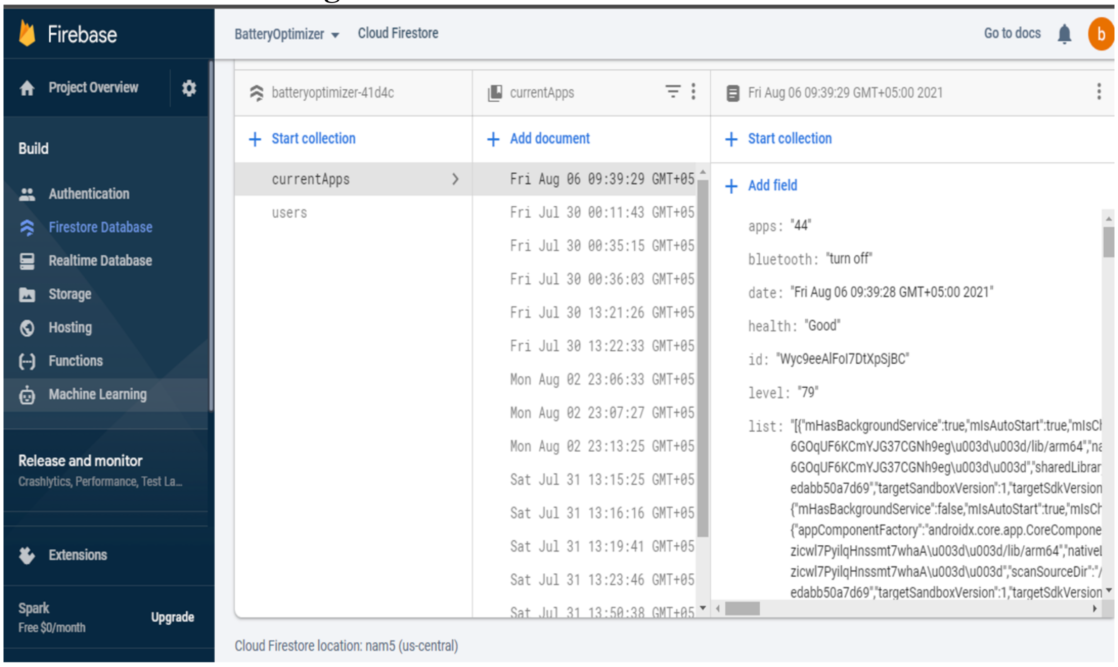


```
currentApps
users

dua@gmail.com
itsshaikhtaha@gmail.com
itssjohn@gmail.com
m@gmail.com
m@m.com
veena@gmail.com

email: "dua@gmail.com"
id: "FExalN7PaF30sHSSIHru"
name: "dua"
pass: "12345"
```

Figure 7: Firebase-Individual User Details



```
currentApps
users

Fri Aug 06 09:39:29 GMT+05
Fri Jul 30 00:11:43 GMT+05
Fri Jul 30 00:35:15 GMT+05
Fri Jul 30 00:36:03 GMT+05
Fri Jul 30 13:21:26 GMT+05
Fri Jul 30 13:22:33 GMT+05
Mon Aug 02 23:06:33 GMT+05
Mon Aug 02 23:07:27 GMT+05
Mon Aug 02 23:13:25 GMT+05
Sat Jul 31 13:15:25 GMT+05
Sat Jul 31 13:16:16 GMT+05
Sat Jul 31 13:19:41 GMT+05
Sat Jul 31 13:23:46 GMT+05
Sat Jul 31 13:50:38 GMT+05

apps: "44"
bluetooth: "turn off"
date: "Fri Aug 06 09:39:28 GMT+05:00 2021"
health: "Good"
id: "Wyc9eeAlFoI7DtXpSjBC"
level: "79"
list: "[{"mHasBackgroundService":true,"mIsAutoStart":true,"mIsCl
      6GOqUF6KCmYJG37CGNh9eg\u003d\u003d/lib/arm64","na
      6GOqUF6KCmYJG37CGNh9eg\u003d\u003d","sharedLibrar
      edabb50a7d69","targetSandboxVersion":1,"targetSdkVersion
      {"mHasBackgroundService":false,"mIsAutoStart":true,"mIsCh
      {"appComponentFactory":"androidx.core.app.CoreCompone
      zicwl7PyilqHnssmt7whaA\u003d\u003d/lib/arm64","nativel
      zicwl7PyilqHnssmt7whaA\u003d\u003d","scanSourceDir":"/
      edabb50a7d69","targetSandboxVersion":1,"targetSdkVersion
```

Figure 8: Firebase-Individual User Cleanup

Conclusion:

This research has effectively applied an approach to increase battery power in mobile devices, which delivers a solution to continuous battery depletion amid improvements in mobile technology. Our innovation is intelligently spotting power-consuming components in a mobile phone and applying the code-offloading method to run the computation process on a cloud server. This approach is achieved by the convergence of mobile computing with cloud computing to ease the battery usage of the mobile itself. The proposed optimizer for mobile phone applications leads to considerable power savings, extending battery life, and improving processing capability. This tactic has proven crucial in solving the urgent problem of background processing and power depletion on mobile devices, giving consumers a battery performance that is more effective and lasts longer.

We reported on manual, small-scale feasibility research and highlighted several potential barriers to validate the implication of bandwidth and power consumption optimization in MC hybrid systems. Firebase operates in an event-driven manner, calling the thread that started its on-event handler whenever an event occurs. From the perspective of scalability, using Firebase or other similar systems as middleware might not be the best option. The callback from the Firebase handler will be stopped owing to an intrinsic feature of Android systems if the waiting thread is the main thread to retrieve the results from the cloud. The alternative is to utilize socket-based middleware, which operates in a suspend-migrate-receive-resume manner and generates appropriate responses based on the sent request attributes within a timeframe. It is essential to note that since the study is centered on Mobile Cloud Computing (MCC), the availability and reliability of network connections may have an impact on its efficacy. Users in places with inadequate network coverage may experience limitations in user experience and face challenges in accessing cloud resources. In upcoming phases, when implementing the suggested application, we might conduct a survey to gather feedback. This survey will be useful in incorporating recommendations for data format, storage, and maintenance from various users. Additional enhancements can be made by incorporating extra features like personalized power-saving configurations or widgets for real-time monitoring. In forthcoming research endeavors, Exploring and evaluating different optimization algorithms, including the integration of machine learning, can enhance the system's adaptability to user behavior. Additionally, analysis of energy consumption patterns, and evaluating the environmental impact in terms of green computing contribute to the broader applicability and sustainability of the proposed solution.

References:

[1] H. K. Shaikha and A. B. Sallow, “Mobile Cloud Computing: A Review,” Acad. J. Nawroz Univ., vol. 6, no. 3, pp. 129–134, Aug. 2017, doi: 10.25007/AJNU.V6N3A96.

[2] S. C. Toader, R. I. Ciobanu, and C. Dobre, “Smart Computation Offloading for Mobile Clouds,” 2023 IEEE Int. Conf. Pervasive Comput. Commun. Work. other Affil. Events, PerCom Work. 2023, pp. 62–67, 2023, doi: 10.1109/PERCOMWORKSHOPS56833.2023.10150282.

[3] C. Nigam, G. Sharma, and E. Menghani, “Comprehensive Review and Analysis on Mobile Cloud Computing and Users Offloading using Improved Optimization Approach for Edge Computing,” Int. J. Intell. Syst. Appl. Eng., vol. 10, no. 1s, pp. 234–242, Oct. 2022, Accessed: Feb. 06, 2024. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/2286

[4] P. Eastham, A. Sharma, U. Syed, S. Vassilvitskii, and F. Yu, “Learning Battery Consumption of Mobile Devices,” 2016.

[5] M. Ali, J. M. Zain, M. F. Zolkipli, and G. Badshah, “Mobile cloud computing & mobile battery augmentation techniques: A survey,” 2014 IEEE Student Conf. Res. Dev. SCOReD 2014, Mar. 2014, doi: 10.1109/SCORED.2014.7072944.

[6] M. R. Nakhkash, T. N. Gia, I. Azimi, A. Anzanpour, A. M. Rahmani, and P. Liljeberg, “Analysis of performance and energy consumption of wearable devices and mobile gateways in IoT applications,” ACM Int. Conf. Proceeding Ser., vol. Part F148162, pp. 68–73, 2019, doi: 10.1145/3312614.3312632.

[7] “View of A Systematic Survey of Simulation Tools for Cloud and Mobile Cloud Computing Paradigm.” Accessed: Feb. 06, 2024. [Online]. Available: https://jisrc.szabist.edu.pk/ojs/index.php/jisrc/article/view/40/30

[8] V. P. Estamsetty, “Cloud Computing , Mobile Cloud Computing and its Comparative Study,” no. January, 2021, doi: 10.13140/RG.2.2.30812.41601.

[9] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “CloneCloud: Elastic execution between mobile device and cloud,” EuroSys’11 - Proc. EuroSys 2011 Conf., pp. 301–314, 2011, doi: 10.1145/1966445.1966473.

[10] Y. Hmimz, M. El Ghmary, T. Chanyour, and M. O. Cherkaoui Malki, “Computation offloading to a mobile edge computing server with delay and energy constraints,” 2019 Int. Conf. Wirel. Technol. Embed. Intell. Syst. WITS 2019, Apr. 2019, doi: 10.1109/WITS.2019.8723733.

[11] S. Cao, X. Tao, Y. Hou, and Q. Cui, “An energy-optimal offloading algorithm of mobile computing based on HetNets,” 2015 Int. Conf. Connect. Veh. Expo, ICCVE 2015 - Proc., pp. 254–258, Apr. 2016, doi: 10.1109/ICCVE.2015.68.

[12] D. J. S. Raj, “Improved Response Time and Energy Management for Mobile Cloud Computing Using Computational Offloading,” J. ISMAC, vol. 2, no. 1, pp. 38–49, Mar. 2020, doi: 10.36548/JISMAC.2020.1.004.

[13] M. A. Elgendy, A. Shawish, and M. I. Moussa, “MCACC: New approach for augmenting the computing capabilities of mobile devices with Cloud Computing,” Proc. 2014 Sci. Inf. Conf. SAI 2014, pp. 79–86, Oct. 2014, doi: 10.1109/SAI.2014.6918175.

[14] Y. Sun, J. Wu, L. Chen, T. Liu, M. Yao, and W. Sun, “Latency optimization for mobile edge computing with dynamic energy harvesting,” Proc. - 2019 IEEE Intl Conf Parallel Distrib. Process. with Appl. Big Data Cloud Comput. Sustain. Comput. Commun. Soc. Comput. Networking, ISPA/BDCloud/SustainCom/SocialCom 2019, pp. 79–83, Dec. 2019, doi: 10.1109/ISPA-BDCLOUD-SUSTAINCOM-SOCIALCOM48970.2019.00022.

[15] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, and M. S. Hossain, “Intelligent task prediction and computation offloading based on mobile-edge cloud computing,” Futur. Gener. Comput. Syst., vol. 102, pp. 925–931, Jan. 2020, doi: 10.1016/J.FUTURE.2019.09.035.

[16] A. Ali et al., “An Efficient Dynamic-Decision Based Task Scheduler for Task Offloading Optimization and Energy Management in Mobile Cloud Computing,” Sensors 2021, Vol. 21, Page 4527, vol. 21, no. 13, p. 4527, Jul. 2021, doi: 10.3390/S21134527.

[17] A. Akbar and P. R. Lewis, “Towards the optimization of power and bandwidth consumption in mobile-cloud hybrid applications,” 2017 2nd Int. Conf. Fog Mob. Edge Comput. FMEC 2017, pp. 213–218, Jun. 2017, doi: 10.1109/FMEC.2017.7946433.

[18] N. Dange, K. Devadkar, and D. Kalbande, “Scheduling of task in collaborative environment using mobile cloud,” Proc. - Int. Conf. Glob. Trends Signal Process. Inf. Comput. Commun. ICGTSPICC 2016, pp. 579–583, Jun. 2017, doi: 10.1109/ICGTSPICC.2016.7955367.

[19] I. Kaur, S. Sharma, and M. Arora, “Research Paper on Enhanced Battery for Android Phones using the Power of Cloud through Data Synchronization,” 2014.

[20] A. Wajid, N. Nigar, S. Islam, and M. K. Shahzad, “A SURVEY ON MOBILE CLOUD COMPUTING PROBLEMS AND SOLUTIONS,” Pak. J. Sci., vol. 75, no. 1, pp. 71–77, Mar. 2023, doi: 10.57041/PJS.V75I1.824.

[21] B. Assistant Professor, “Mobile Cloud Computing: Implementation using Android and Firebase API,” Int. J. Creat. Res. Thoughts, vol. 6, no. 2, pp. 2320–2882, 2018, Accessed: Feb. 06, 2024. [Online]. Available: www.ijcrt.orgwww.ijcrt.org

[22] A. B. Semma, M. Ali, M. Saerozi, Mansur, and Kusrini, “Cloud computing: google firebase firestore optimization analysis,” Indones. J. Electr. Eng. Comput. Sci., vol. 29, no. 3, pp. 1719–1728, Mar. 2023, doi: 10.11591/IJEECS.V29.I3.PP1719-1728.