Rural Digital Empowerment: A Flutter Application Framework for Self Help Group

Naga Sushanth Kumar Vuppala,
Pappala Kumar Aditya
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Amritapuri, India
amenu4aie20171@am.students.amrita.edu,
amenu4aie20155@am.students.amrita.edu,

Sruthy Anand, Ramesh Guntha
Center for Wireless Networks & Applications (WNA)
Amrita Vishwa Vidyapeetham
Amritapuri, India
sruthyanand@am.amrita.edu,
rameshg@am.amrita.edu

***Abstract*—Mobile applications have become an integral part of our modern society, reaching various aspects of our lives. The rural community is still a way back in this digital era regarding technology usage. This project aims to improve the digital skills and utilization of digital services among women's Self-Help Groups (SHGs) by offering software and allowing them to use smartphones and apps. The success of any technology created for the rural community is how readily the users embrace and use it. This initiative will help bridge the digital divide and empower women in these groups to manage their SHG activities effectively. The application's users are women from different rural areas in India with varying literacy and digital proficiency levels. The software utilized a multi-level client-server architecture using Node.js and Flutter frameworks. The software's performance testing shows it can handle a large user base.**

***Index Terms*—Rural Development, Self-Help Groups, User Interface, Mobile Application, Performance Testing, Flutter, Cross-Platform.**

I. Introduction

Rural development relates to enhancing the living standards, education opportunities, healthcare access and economic prosperity of those residing in rural regions of India. Digital empowerment and technology can act as a catalyst for the rural development. The growing use of mobile phones by the rural community presents a significant opportunity to improve rural communities by empowering them with digital technology use. Digital empowerment through mobile applications holds the potential to bring about substantial positive changes in community life. For example, the rise of banking and digital payment systems, like the Unified Payments Interface (UPI), has significantly transformed how people in rural areas use digital technologies. UPI, with its accessibility through a mobile phone number and PIN, has made financial services readily available, particularly beneficial for those with limited access to traditional banking services [1].

The increased use of digital payments in rural areas has led to the growth of the digital economy, creating new opportunities for businesses and entrepreneurs and improving livelihoods [2]. The concept of digital empowerment has the potential to aid in substantial transformations in rural community life by equipping them with the latest technologies and applications that can enhance their quality of life. Moreover, the concept of digital empowerment, aligned with the Indian government's Digital India plan, seeks to advance digitization, create jobs, enhance infrastructure, and support small companies.

Women empowerment is a significant aspect of rural well-being that has been addressed through Self-Help Groups (SHGs) in India. SHGs have emerged to empower women in rural areas by providing an additional avenue for generating income and fostering collaborative efforts within their communities [3]. Generally SHG initiative has been instrumental in helping women become self-sufficient and improve their quality of life. This comprehensive approach aims to increase women's financial stability, academic performance, and skill set [4]. AmritaSREE, an initiative launched by the Mata Amritanandamayi Math, is a notable example focusing on women's socio-economic development through the formation of SHGs [5]. AmritaSREE emphasizes financial inclusion and economic empowerment by bringing women together in SHGs, promoting collective savings, micro-credit, and entrepreneurship [3].

Researchers have proposed an participatory Internet of things solution to digitize the SHG financial management to increase the transparency of the SHG [3]. Prior research also suggested the need of consensus and a hybrid chatbot [6] for supporting the SHG. In this paper, authors focus highlighting the software implementation and the design strategy with details of the client-server architecture used to implement the software. We have used Flutter as front-end framework as it provide a single codebase that can be used to deploy applications on multiple platforms, including iOS, Android, web, and desktop, saving development time and resources [7]. Flutter is also an attractive option for front-end development due to its performance, efficiency, and versatility. We have used Node.js for server side coding as it is modular, smaller, and simpler. Node.js performs better than other alternatives such as PHP and Python-Web in terms of how many requests the server can handle and how quickly the server can answer a query [8].

The remainder of this paper is structured as follows. Section II discusses the related works in this research. Section III discusses the SHG functioning and application design stratergy. Section IV discusses the UI design aspect and implementation details of Flutter UI. Section V details the implementation of server-side coding details for the application. The software development methodology equipped to build and the application have been detailed in Section VI, Concluding with Future Work in section VII.

II. Related Works

The literature explores various fields such as the impact of software solutions in the context of rural development, different proposals of software solutions to empower the SHGs, understanding the development details of Flutter and Node.js frameworks. The paper [9] presents a case study whereby a mobile platform empowers rural people in three different aspects like structural, psychological and resource empowerment. Several smartphone applications designed to empower the rural population in various fields such as to deliver timely information, to provide market information and for healthcare services [10]. This assisted them to accomplish some growth in their living conditions. The research focused on rural villages in Cornwall, UK, had a recent access to broadband through a regional program [11]. The study revealed that while offering technology access is crucial for digital inclusion, it's equally important to empower rural communities to leverage their social structures. This study [11] provides the significance of digital inclusion in rural development.

The authors of [6] talk on how SHGs might benefit from mobile applications and adaptive user interfaces as well as possible social benefits. Another area where there is clear promise to improve the accessibility, cost, and availability of essential services for this group is chatbots [6]. The transition from manual to digital ledger maintenance through a mobile application will improve overall system performance, security, and transparency. Digitization using databases retains detailed group transaction information and offers members convenient digital access to ledgers, facilitating anytime transaction checks [5]. The study [12] introduce CAM, a novel framework designed for easy mobile computing applications to the needs of rural communities in the developing world. CAMForm is simple yet efficient design for the submission of microfinance loan applications.

Many previous studies demonstrate that lower levels of formal education and literacy will make it challenging for many people to receive the advantage of such solutions [13]. The baseline level of technological knowledge among older generations is noticeably lower. As a result, they discover that it is getting harder for them to adjust to the quickly evolving surroundings, technology, and trends. Addressing diverse user groups with varying age, literacy levels, and mobile usage patterns necessitates a robust approach, wherein incorporating local language text, supplemented by images and voice support, proves to be a better idea [3].

Generally developers opt for cross-platform frameworks for develping software's due to their convenience, allowing them to write code once and deploy it on both iOS and Android simultaneously. This approach saves time and effort by eliminating the need to acquire distinct skill sets for each platform [14]. The flexible and adaptable widget-based UI framework of Flutter makes it possible to create simple user interfaces. Flutter also has a large library of widgets and a thriving community, both of which speed up the development process [15]. Leveraging the Dart programming language, Flutter ensures fast and smooth graphics rendering across various platforms [16] [17]. The framework's approach to building user interfaces accelerates rendering by only updating widgets when changes occur [18].

The research conducted in [8] illustrates Node.js outperforming PHP and PythonWeb in terms of both server request handling capacity and the speed at which the server can respond to a MySQL SELECT query. Node.js offers a technical advantage with its event-driven architecture, enhancing scalability and efficiency by enabling asynchronous handling of events and I/O operations [19]. Beyond technical advantages, Node.js attracts interest for fostering a more human approach to programming, drawing developers who appreciate its simplicity and community support [20]. Applications will allow users to provide their login information once, and in exchange receive a encrypted string of random characters for a unique token based authentication system [21]. This token will be used to provide access to application, as the digital token acts as confirmation that the user has already been authenticated.

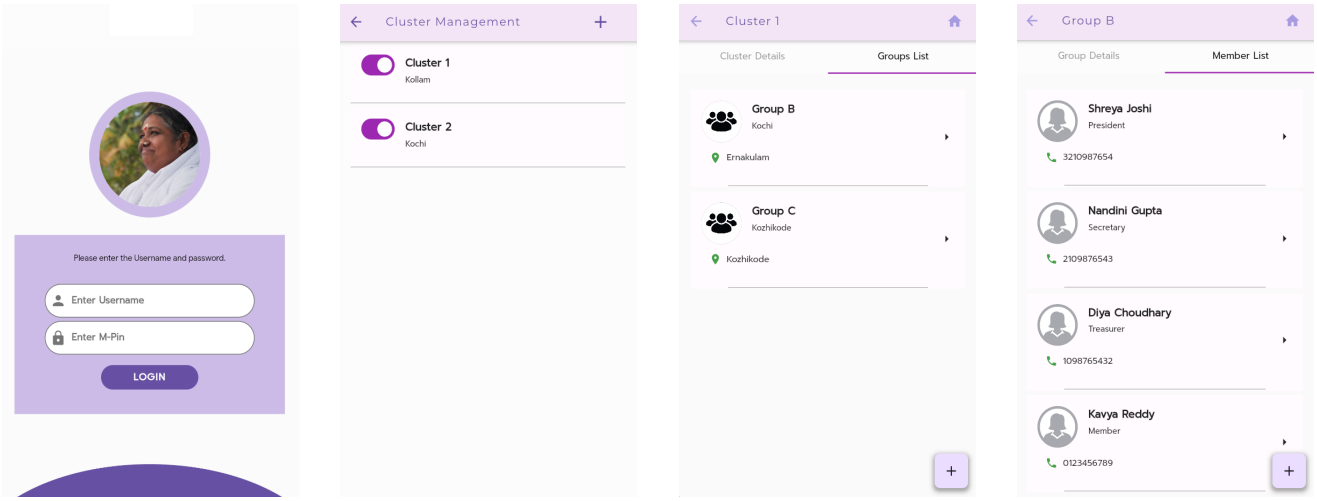
The existing works highlight the impact of digital solutions, particularly mobile applications, on rural communities. The focus ranges from empowering SHGs in rural areas through innovative mobile platforms to the potential benefits of chatbots and digitization in improving essential services. The studies show the importance of diverse user groups with different literacy levels. Additionally, the choice of development frameworks, such as Flutter for cross-platform in the client side and technical advantages of Node.js as server is good as per the existing work.

III. Software Design for SHG

This section details the SHG functioning, the design strategy we used and the client-server technology we used for the development. Details of the client architecture, UI design and modularity we adopted have also been discussed.

A. SHG Functioning

AmritaSREE SHGs operate as a community-based initiative with a dedicated mission to empower women through economic support. The functioning of SHG involves rural women living nearby form a group of 10-20 members. Several groups form a cluster which is managed by the cluster administrator. The collective pool of all SGHs within a cluster is then distributed to the SHG users as a collateral free loan. By



(a) Login screen (b) Cluster Details Screen (c) SHG List within a Cluster (d) Member List within a SHG

Fig. 1: Screenshots of the Application indicating icon based UI

enabling women to receive collateral free loans, enhance family well-being, and participate in decision-making processes, AmritaSREE contributes to community development through collaborative efforts and increased financial support within communities [22]. AmritaSREE operates by forming SHGs comprised of 10-20 women who meet weekly to collectively save modest amounts of money. The savings mechanism involves each member contributing to a common pool, from which the cluster admin can provide loans to individual members at a lesser interest rate. AmritaSREE supports SHGs in achieving financial independence from traditional banks by facilitating the formation of clusters. Clusters, comprising various SHGs, pool their economic resources, enabling mutual lending among SHGs without relying on external banking institutions. This approach enhances financial resilience and promotes a self-sustaining model within the community. Designing a software with different user characteristics requires a design strategy which has been detailed in the upcoming section.

B. Application Desing Strategy

Designing a mobile application that can be effectively used and adapted by rural women of all age groups requires a thoughtful design strategy. Given the diversity in digital literacy and age demographics, it is crucial to design a linear application workflow, ensuring a step-by-step navigation approach. This is particularly important for users who might be new to mobile applications, especially among the elderly population. Ensuring a user-friendly interface, avoiding confusing navigation and clutter is a critical aspect in the application development. Utilization of symbols, words, and visuals familiar to the diverse target audience. The UI should be simple, self-explanatory [3]. Enabling users to select their preferred language upon the app's initial launch, with easy access in settings will help more users get adapted to the application. Translate all app text, including menus, buttons, labels, and content, ensuring accuracy and cultural sensitivity. Implement a seamless language-switching feature within the app, this will be beneficial for households with multiple users and diverse language preferences. A glimpse of the application developed is shown in the Figure. 1 which reflects the design stratergy mentioned.

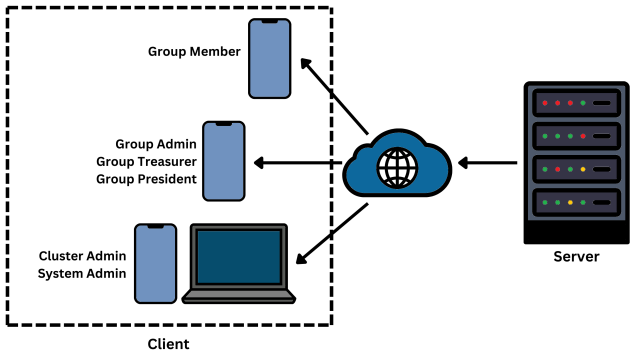


Fig. 2: Client-Server architecture for the Application

IV. Design Aspect of the Client Side

The software is developed using the client-server architecture, inspired from the work [23]. The client-server architecture, illustrated in Fig. 2, can serve as the framework for a cross-platform application. The client side of the application is developed by leveraging the Flutter technology that is backed by Google. This application supports Mobile (Android, iOS), Web, and Desktop (Windows and macOS) through the utilization cross-platform development capabilities of Flutter. This allow developers to write code for the application once, using a single codebase, and deploy it simultaneously on various platforms. Unlike other technologies, Flutter builds its applications in a native environment, which improves performance and facilitates easy integration with device components

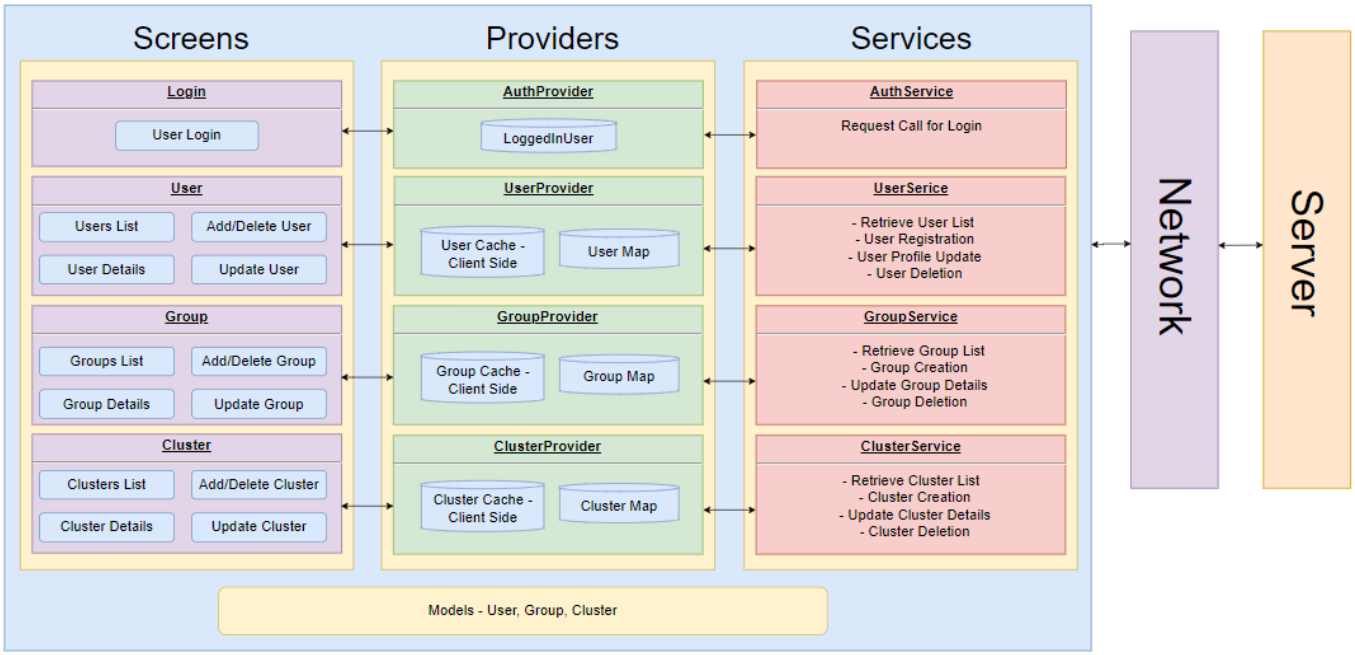


Fig. 3: Client Side Architecture for the Application

like the camera and GPS. Also, flutter is a react based development framework which makes it flexible to develop the applications. By organizing the code into modular widgets, classes and utilizing Flutter's rich widget library, we can create a maintainable and flexible UI for the application. The state management and widget composition make it easier to extend and enhance the application's functionality in the future.

A. *Flutter SDK Kit*

Widgets are the fundamental building blocks of Flutter UI development. They represent the UI components and layout elements that make up the app's user interface. Flutter provides various in-built widgets such as buttons and text fields to complex layouts and animations. Flutter also allows developers to build either low-level or high-level custom widgets based on UI enhancement requirements. The widgets are organized in a tree structure, forming the visual hierarchy of the app. They are reusable, enabling us to create rich, interactive, and responsive user interfaces efficiently. Whether designing a simple screen or a complex application, understanding and using Flutter widgets effectively is essential for creating visually appealing and functional apps.

- StatefulWidget and State: This allows the screen to have mutable state that can change during the app's life cycle.
- Form and GlobalKey¡FormState¿: The Form widget is used to create a form with required fields. It is associated with a GlobalKey¡FormState¿, which is used for form validation and saving form data.
- TextFormField: TextFormField widgets are used for capturing user input. They include features like validation, input formatting, and saving user input.
- SharedPreferences: Flutter plugin that provides a way to persistently store simple data. It's commonly used for storing and retrieving user preferences or small amounts of data between app sessions.
- Material, InkWell, Container: These widgets are used for creating interactive elements like buttons (InkWell) with material design styling (Material). Containers are used for layout and spacing purposes.
- ListView: ListView is used to create a scrollable list of widgets.
- BuildContext: BuildContext is used to access the context within which widgets are built. It's essential for navigation and widget composition.
- AppBar: AppBar widget represents the app bar at the top of the screen. It contains a title and other neccessary actions.
- CircleAvatar: CircleAvatar widget creates a circular avatar for displaying user images. It's commonly used for profile pictures.

B. *Modularity*

In software development, modularity is a design that divides a large, complex system into smaller, independent modules or components. Each module performs a certain set of functions and can be built, tested, and maintained independently. Each module has its own classes based on the functionalities, which can modified without creating creating bugs in other parts of the application. Modularity promotes code re-usability, maintainability, and scalability, making it easier to manage and enhance software systems. It also supports collaboration among developers as they can work on separate modules simultaneously. The each horizontal level of the Figure. 3

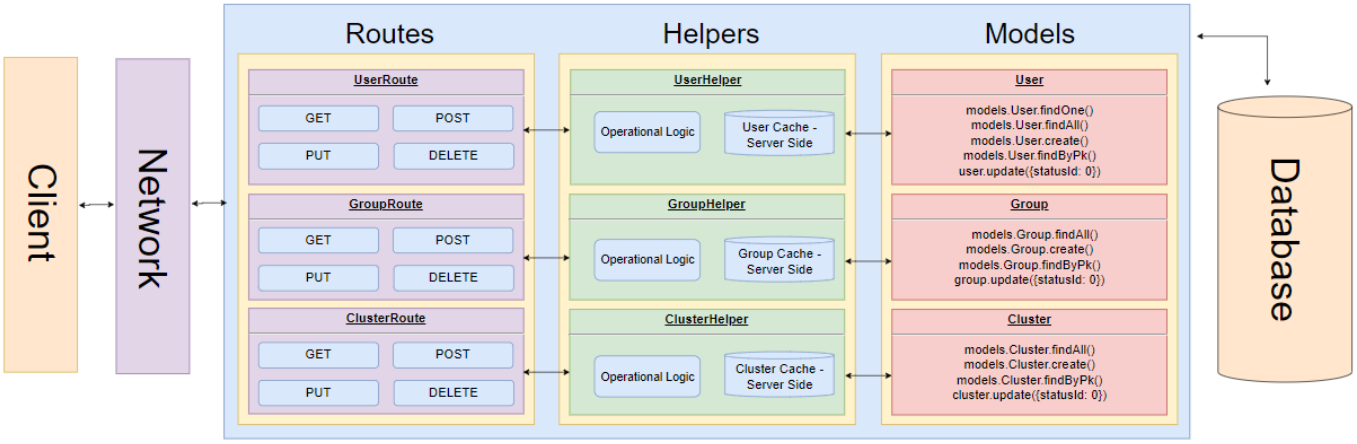


Fig. 4: Server architecture for the Application

its structured into a separate module based on the functional aspect like User Module, Group Module, and Cluster Module.

C. Layered Client Architecture

The client side architecture of the mobile application is shown in Fig. 3. The architecture is built on three levels namely Screen, Providers, and Services. Models like user model, group model, cluster model are similar to user-defined data types which are used carry the data required within the system. This architecture makes it easier to maintain and test different parts of application independently. Screens are represented by Widgets. Each screen is built using the build method of a widget class. Screens are responsible for user interaction, manage the UI elements, and display of required data. Providers are responsible for managing the application state and Client Side Cache. They are used to share data and functionalities across different parts of the application. Services are responsible for interacting with APIs or databases. They execute the implementation details of fetching or sending data. Services are responsible for making HTTP requests in the application.

D. User Authorization

The client-side coding ensures that users are restricted from accessing specific functionalities based on the permissions assigned to their roles within the system, achieved by withholding the display of certain buttons. This ensures, strict access to different functionalities within the system. The similar type of user authorization is also performed in the Server side to restrict the data from other users with different roles.

V. Server Side Coding

Server-side coding is integral to mobile application development, handling data, logic, and functionality. It ensures secure storage, retrieval, and processing of data while managing real-time exchanges between clients. Node.js is a preferred choice due to its event-driven architecture, scalability for concurrent connections, and a vast ecosystem with an active community. It excels in real-time requirements, making it suitable for mobile apps that demand responsiveness and scalability. Express.js, a minimalist web framework for Node.js, further streamlines server-side coding in mobile application development by providing a robust set of features for routing, middleware integration, and handling HTTP requests and responses efficiently. Node.js focus on performance improvements, including worker threads and a refined V8 JavaScript engine, enhances execution speed and reduces latency [24]. Features like worker threads and enhancements to the V8 JavaScript engine contributed to better execution speed and reduced latency.

A. Server Architecture

The functionalities in the server side also structured into different modules similar to client side. The architecture for server-side coding has been shown in Fig. 4. The architecture is built on three levels namely Routes, Helpers, and Models. The routes define the endpoints and corresponding handlers for your server. The handler for this route uses a helper function to retrieve data and then responds to the client based on the success or failure of the operation. Routes act as the entry points to your server's functionality. Helpers contain the operational logic and functions that assist in handling data processing and operations. Helpers encapsulate the core functionality of your application and are responsible for performing specific tasks. Models represent the data structure of your application and define how data is stored in the database. Sequelize ORM is used to interact with the database, providing an abstraction over SQL operations.

Hypertext Transfer Protocol Secure (HTTPS) techniques are crucial for safely interacting with web servers and retrieving or transmitting data while developing mobile apps. Four typical HTTPS techniques for creating mobile apps are GET, PUT, POST, and DELETE [25]. In order to retrieve data from a server, such as user profiles, news articles, or product details, mobile apps frequently employ GET queries. Data is sent to the server for processing or storing via the POST technique. A resource on the server can be updated or changed using

the PUT technique. Data is sent to a given resource location, where it is overwritten if it already exists or created if it doesn't. When mobile apps need to update user profiles, change settings, or amend existing information, they use PUT requests. A resource can be requested to be deleted from the server using the DELETE method. It is employed to permanently remove particular data. When users want to delete their accounts, remove content, or carry out any other activity that necessitates the permanent removal of data from the server.

B. User Authentication

Verifying a person's identity through user authentication takes place normally prior to providing them access to a system or application. User will be given a token upon successful login, which is further used to authenticate the access of the user. By ensuring that only authorised users based on different user roles, they may access resources, authentication improves security and shields user data from unauthorised access. It is a crucial part of cyber security across many domains.

C. Performance and Scalability

Performance refers to its ability to execute tasks with minimal time and resource consumption, including reduced memory usage, network data, bandwidth, disk space, CPU usage. Achieving high performance involves optimizing resources to minimize both time and resource usage, aiming for the least possible time required. Strategies such as offloading data to clients or client caching, compression techniques, and server-side caching contribute to enhanced performance. On the other hand, scalability pertains to a system's capability to maintain satisfactory performance even when accommodating a larger number of users or increased workload. The objective is to design the application in such a manner that its performance remains stable even as the number of users or requests continues to grow.

D. Configuration Management

The configurations of the node packages used in building the Server-side programming of the application as shown in Table. I.

TABLE I: Node Packages Information

Node Package	Version	Description
NodeJs	18.6.2	Node.js Version
winston	4.1.0	handling HTTP request logging.
crypto	-	Provides cryptographic functionalities
mysql2	3.1.2	Interact with MySQL databases
sequelize	5.8.7	ORM (Object-Relational Mapping)
cookie-parser	1.4.6	parsing cookies in incoming requests
body-parser	1.20.0	parsing the body of incoming requests

VI. Software Development Methodology

The sections briefs about the development methodology used in the implementation of the application. The pace of development, maintainability and different testing techniques used are discussed in the section.

A. Pace of Development

We have used Agile Methodologies to enhance the development process and ensure the software meets the unique needs of SHG users. Requirements for the application were collected based on the interactions with AmrtiaSREE SHG users. This user-centric, Agile-driven approach ensures that our software meets the immediate requirements of SHGs enhancing community empowering.

B. Maintainability

In agile software development, the functional features of the application keep on developing as staged process with the most essential at the first level continued with development of remaining features. Hence version control is essential for tracking changes in the codebase, collaborating with other developers, and rolling back to previous states if necessary. We have used GitLab as version control system for building this application. A consistent coding standards were used to develop the application such as naming conventions and error handling.

C. Testing Techniques

During the development of the software applications, various testing approaches are employed to enhance software quality. The testing of the system is done to test its performance under different workloads and to check if the designed system is scalable.

1) Testing Objective: Performance testing aids us in evaluating the effectiveness of our implementation strategy, ensuring optimal functional services and user experience. By testing our application, we intend to deliver a robust and high-quality software system.

2) Testing Design: The testing design consists of two distinct approaches to assess the performance of our system. The first method involves measuring the server's response time for handling a single request and transmitting the necessary data. A testing cycle that includes various actions such as user login, retrieval of user, group, and cluster data, and user logout was scripted. In order to evaluate the server's performance under various user loads, the second approach focuses on making concurrent requests based on this testing cycle, with varied numbers of simultaneous users.

3) Testing Environment Setup: For the testing environment setup, the server was executed on our local computer equipped with an AMD Ryzen 7 4800H CPU and 16GB of RAM. The server was simulated with requests using the POSTMAN collection runner, a tool that allows users to execute a series of API requests in either a sequential or concurrent manner to analyze the responses efficiently. To simulate a scenario, 100 distinct clusters were generated, each containing 100 groups, with each group comprising 20 users, resulting in a total of 200,000 users spread across different groups and clusters.

4) Testing Results: As a result of the first approach, the response time for each requests from various user roles was recorded, as shown in Table II. There is a notable spike in response time when retrieving users for the System Admin

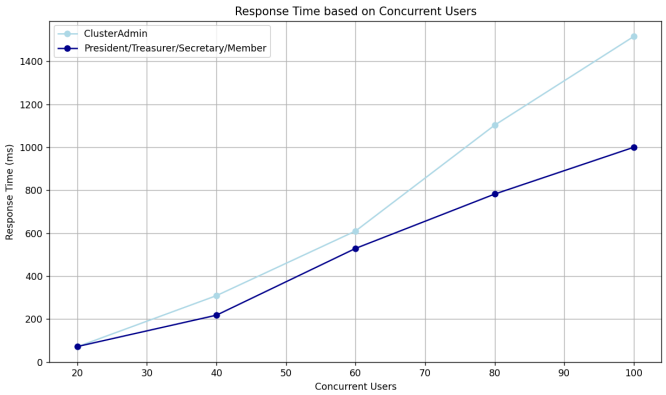


Fig. 5: Response Time for each testing cycle

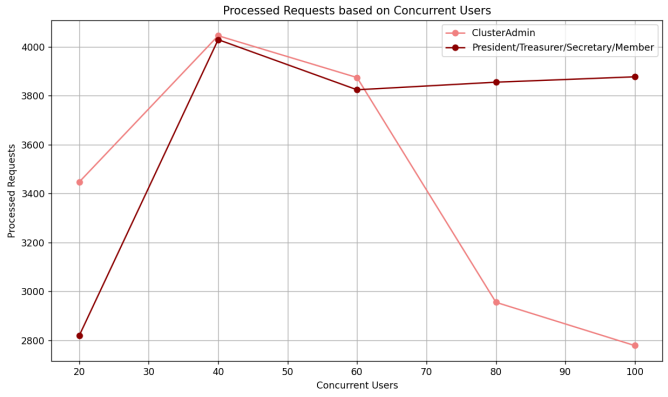


Fig. 6: Number of Processed Requests based on varying concurrent Users

position, as the system needs to fetch all available user data. The analysis of the second approach include the response time for each testing cycle and the number of requests processed within a minute are shown in Fig. 5 and Fig. 6 respectively, when logged in with different user roles. Additionally, the system's RAM usage and CPU utilization were also observed and is shown in Fig. 7. The analysis of the results reveals that the system exhibits stable usage patterns even as the user load increases. This stability suggests that the system possesses scalability and can handle higher user loads without significant degradation in performance. Furthermore, the system appears to operate well under the capacity and is capable for accommodating additional user load.

TABLE II: Response Time(ms) for each Request

Response Time (ms)	System Admin	Cluster Admin	President	Member
Login	13	12	15	12
Edit Password	10	10	10	10
Retrieve Users	577	71	45	48
Create User	17	17	17	17
Update User	24	25	24	24
Delete User	21	21	22	-
Logout	8	7	7	7
Retrieve Groups	47	16	10	10
Create Group	16	17	16	16
Update Group	16	17	16	16
Delete Group	15	15	-	-
Retrieve Clusters	8	7	7	7
Create Cluster	17	-	-	-
Update Cluster	17	16	-	-
Delete Cluster	21	-	-	-

VII. Conclusion

Rural development enhances the living standards of the individuals. The paper plays a key role in understanding the role of digitization in rural development. Our research also looked into existing studies on how software solutions can help improve rural people's living standards. Digitization of rural SHGs will streamline the process and help users achieve digital inclusion. This study helps developers and researchers as we describe the root-level details of server-side and client-side coding in the context of developing mobile applications

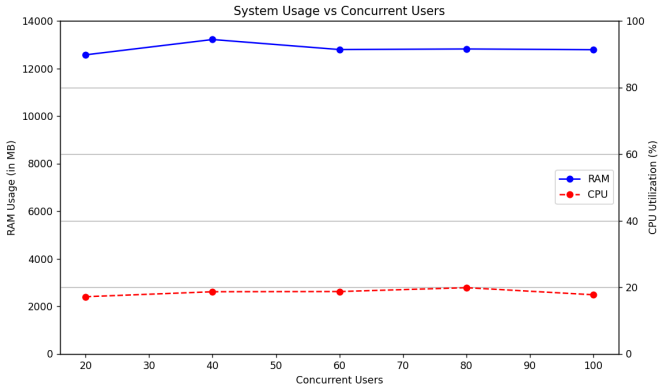


Fig. 7: RAM Usage and CPU utilization based on varying Concurrent Users

for users of various age groups. The usage of Node.js will help developers update multiple data tables through a single request, and the rollback feature can be utilized to revert the state and preserve the original data in case of a failed operation. The software reliability and performance testing results show that the server can incorporate a large user base. Future work for this study involves integrating a payment module for the SHGs. This module facilitates real-time tracking of financial transactions with the SHGs.

Acknowledgment

We would like to thank our beloved Shri. (Dr) Mata Amritanandamayi Devi, our institution's Chancellor for her encouragement, support and guidance.

References

[1] M. Thirupathi, G. Vinayagamoorthi, and S. Mathiraj, "Effect of cashless payment methods: A case study perspective analysis," *International Journal of scientific & technology research*, vol. 8, no. 8, pp. 394–397, 2019.

[2] M. A. and G. Bhat, "Digital payment service in india - a case study of unified payment interface," *International Journal of Case Studies in Business, IT, and Education*, pp. 256–265, 06 2021.

[3] S. Sreeraj, A. Unnikrishnan, K. Vishnu, N. E. Kennith, S. Anand, and M. V. Ramesh, "Empowerment of women self help groups: human centered design of a participatory iot solution," in *2020 IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, 2020, pp. 1–8.

[4] G. Yoganandham, "The consequences of digital india on the emancipation of rural women in vellore district of tamil nadu-an economic analysis."

[5] A. N. KA, A. Rakesh, V. Alok, M. Ananthu, S. Anand, and M. V. Ramesh, "Consensus agreement for secure transactions in self help groups," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021, pp. 1–6.

[6] S. Anand, M. Karthikeya, A. Abhishek Sai, and O. Balamurali, "Multilingual hybrid chatbot for empowering rural women self-help groups in india," in *2023 International Conference for Advancement in Technology (ICONAT)*, 2023, pp. 1–6.

[7] L. Dagne, "Flutter for cross-platform app and sdk development," 2019.

[8] K. Lei, Y. Ma, and Z. Tan, "Performance comparison and evaluation of web development technologies in php, python, and node. js," in *2014 IEEE 17th international conference on computational science and engineering*. IEEE, 2014, pp. 661–668.

[9] L. Ye and H. Yang, "From digital divide to social inclusion: A tale of mobile platform empowerment in rural areas," *Sustainability*, vol. 12, no. 6, p. 2424, 2020.

[10] B. S. Mehta, "Impact of mobile phone on livelihood of rural people," *Journal of Rural Development*, pp. 483–505, 2016.

[11] K. S. Willis, "Making a'place'for icts in rural communities: The role of village halls in digital inclusion," in *Proceedings of the 9th International Conference on Communities & Technologies-Transforming Communities*, 2019, pp. 136–142.

[12] T. S. Parikh and E. D. Lazowska, "Designing an architecture for delivering mobile information services to the rural developing world," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 791–800.

[13] I. Medhi and R. Kuriyan, "Text-free ui: prospects and challenges for ict access," in *Proceedings of the 9th international conference on social implications of computers in developing countries, Sao Paulo, Brazil*, 2007.

[14] "Top 10 Best Cross-Platform App Development Frameworks in 2024 — TechAhead — techaheadcorp.com," https://www.techaheadcorp.com/blog/best-cross-platform-app-development-frameworks/, [Accessed 19-02-2024].

[15] A. Biessek, *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*. Packt Publishing Ltd, 2019.

[16] A. Praveen, K. Nanda, N. Rajith, N. Giriraj, R. Radhika, N. Mahesh, K. Vishnu, T. Anjali, and S. Sarath, "Conference room booking application using flutter," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 0348–0350.

[17] A. Anand, S. Nishanth, P. Vamsi Krishna, S. Krishna, and T. Anjali, "Ally - a crowdsourced distress signal app," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 502–506.

[18] N. Kuzmin, K. Ignatiev, and D. Grafov, "Experience of developing a mobile application using flutter," in *Information Science and Applications: ICISA 2019*. Springer, 2020, pp. 571–575.

[19] D. Herron, *Node. js Web Development: Server-side web development made easy with Node 14 using practical examples*. Packt Publishing Ltd, 2020.

[20] M. Casciaro and L. Mammino, *Node. js Design Patterns: Design and implement production-grade Node. js applications using proven patterns and techniques*. Packt Publishing Ltd, 2020.

[21] P. Pant, A. S. Rajawat, S. Goyal, P. Bedi, C. Verma, M. S. Raboaca, and F. M. Enescu, "Authentication and authorization in modern web apps for data security using nodejs and role of dark web," *Procedia Computer Science*, vol. 215, pp. 781–790, 2022.

[22] "AmritaSREE - AmritaSREE Self Help Groups — amritasree.com," https://www.amritasree.com/, [Accessed 07-03-2024].

[23] R. Guntha, S. N. Rao, H. Muccini, and M. V. Ramesh, "Rapid yet robust continuous delivery of software for disaster management scenarios," *IEEE Software*, vol. 38, no. 4, pp. 104–113, 2020.

[24] F. R. Cogo, G. A. Oliva, C.-P. Bezemer, and A. E. Hassan, "An empirical study of same-day releases of popular packages in the npm ecosystem," *Empirical Software Engineering*, vol. 26, no. 5, p. 89, 2021.

[25] "The 5 essential HTTP methods in RESTful API development — TechTarget — techtarget.com," https://www.techtarget.com/searchapparchitecture/tip/The-5-essential-HTTP-methods-in-RESTful-API-development, [Accessed 17-09-2023].