# CareConnect-Desktop-Caretaker-Management-System-Application

Palwe Priyanka D[1], Shruti Thube[2], Rutuja Awate[3], Pranali Kandekar[4], Ragini Lanke[5].
[1]*Assistant Professor. Sau. Sundarbai. Manik. Adsul. Polytechnic, Chas, Ahilyanagar, India*
[2,3,4,5]*Students. Sau. Sundarbai. Manik. Adsul. Polytechni, Chas, Ahilyanagar, India*

***Abstract*—This paper presents a desktop-based caretaker management system named Care Connect, developed using HTML, CSS, JavaScript, Electron.js and Node.js. The system aims to provide a structured platform for managing caretaker services for elderly people, children, and dependent individuals. It enables users to register as care seekers or caretakers, raise care requests, assign caretakers through an admin module, and track care status. The application also supports mock payment handling and feedback collection to complete the service lifecycle. The proposed system simplifies caretaker coordination through an offline-capable desktop solution. Experimental usage demonstrates that the system operates efficiently and provides a user-friendly interface suitable for real-world caretaker management scenarios.**

## I. INTRODUCTION

Caretaker services are essential for supporting elderly individuals, children, and patients who need continuous assistance. Managing caretaker assignments manually is often slow, error-prone, and inefficient. As demand for organized caretaker services increases, a simple and reliable management system becomes necessary. Desktop-based applications offer stability and ease of use, making them suitable for small institutions and households. This project presents Care Connect, a desktop caretaker management system that streamlines registration, care request handling, assignment, and monitoring. It provides role-based access for admin, caretaker, and care seeker users, reducing paperwork, improving transparency, and enabling coordination for academic applications use.

## II. RELATED WORK

Caretaker and service management systems have been explored in various forms, including web-based platforms and mobile applications. Several studies focus on healthcare management systems that support patient records, appointment scheduling, and staff coordination. Existing solutions often rely on web technologies and require continuous internet connectivity, which may not be suitable for all environments. In recent years, desktop applications built using frameworks like Electron.js have gained popularity due to their cross-platform support and integration of web technologies. Research also highlights the effectiveness of role-based systems for managing users and workflows in service-oriented applications. However, many existing systems are complex and not tailored for small-scale caretaker management. The proposed Care Connect system addresses these limitations by offering a lightweight, desktop-based solution with clearly defined modules for caretaker assignment, status tracking, and feedback management.

## III. PROPOSED ALGORITHM

The proposed system integrates user authentication, care request management, caretaker assignment, service tracking, and feedback handling into a single desktop-based workflow. The algorithm is designed to ensure smooth coordination between care seekers, caretakers, and the administrator using an Electron.js desktop environment and Node.js backend services.

Algorithm Steps

1. Initialize the Care Connect desktop application and load system modules.
2. Display login or registration interface based on user selection.
3. Authenticate user credentials and identify role (Admin, Caretaker, Care Seeker).
4. Redirect user to the respective role-based dashboard.
5. Care seeker submits a new care request with required details.
6. Admin reviews pending care requests and assigns a suitable caretaker.
7. Caretaker accepts the assignment and updates care status.
8. System tracks care progress until completion.
9. Record payment status and collect feedback to close the request.

Key Characteristics

- Role-based access control for secure system usage.
- Centralized caretaker assignment and monitoring.
- Real-time care status updates for transparency.
- Simple desktop-based implementation suitable for academic use.

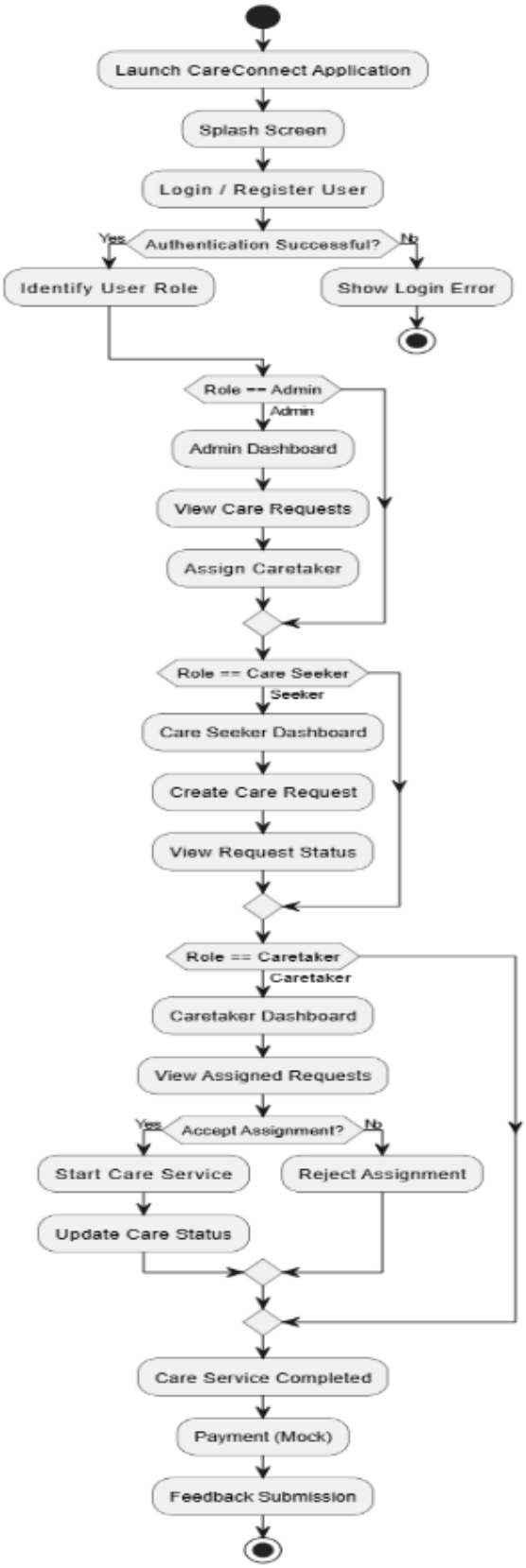


Fig 1.1: -Flowchart Diagram

## IV. SIMULATION RESULT

The proposed Care Connect – Desktop Caretaker Management System was implemented using Electron.js in a desktop environment, with HTML, CSS, and JavaScript for the user interface and Node.js with Express.js for backend logic. The system was tested to evaluate its functionality, usability, and reliability across different user roles including Admin, Caretaker, and Care Seeker. The simulation focused on verifying authentication, care request processing, caretaker assignment, and status tracking operations.

Key Observations:

1. User Authentication: The system successfully authenticated users based on role selection, ensuring secure access to role-specific dashboards without unauthorized access.
2. Care Request Handling: Care seekers were able to create care requests with required details, and all submitted requests were correctly stored and displayed with a pending status.
3. Caretaker Assignment: The admin module efficiently assigned suitable caretakers to pending requests, and assignment updates were immediately reflected in the caretaker dashboard.
4. Care Status Tracking: Caretakers updated care status in real-time (Started, In Progress, Completed), allowing care seekers and admin users to monitor service progress accurately.
5. System Reliability: Continuous testing showed stable performance without application crashes or data loss, demonstrating the system's reliability for desktop-based caretaker management.

Figures:

- Figure 1.1: Flowchart illustrating the overall workflow of the Care Connect desktop application, including role-based authentication, and service completion.
- Figure 1.1: Screenshot of the Care Connect user interface showing role-based dashboards for Admin, Care Seeker, and Caretaker users.
- Figure 1.1: Care status tracking and feedback interface displayed after successful completion of caretaker services.

The flowchart and interface screenshots confirm that the proposed Care Connect system efficiently manages caretaker services through a structured and role-based workflow. The system provides reliable request handling, transparent service tracking, and user-friendly interaction, making it suitable for academic projects and practical caretaker management applications.

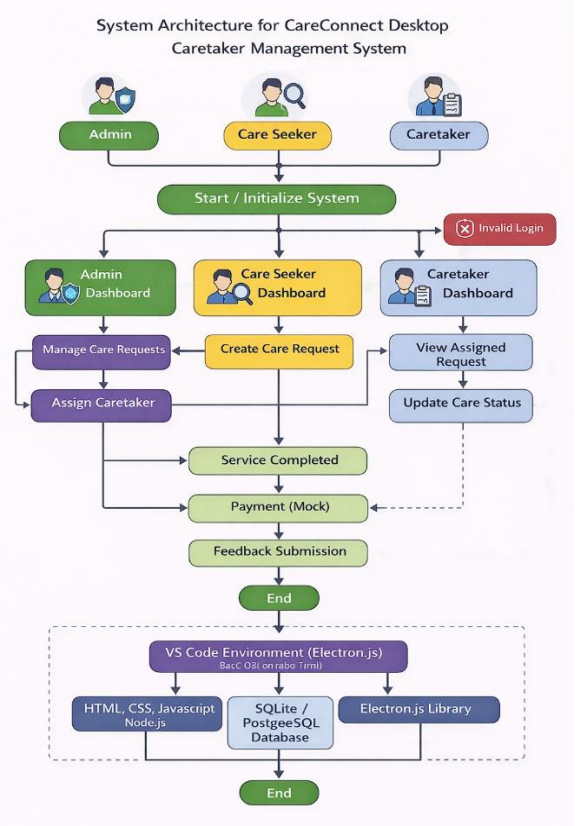


Fig 1.2: -System Architecture Diagram

## V. FUTURE WORK

Future Work

The Care Connect system can be further enhanced with the following improvements:

1. Advanced Care Matching: Implementing intelligent algorithms to automatically suggest the most suitable caretaker based on skills, availability, and care type.
2. Mobile Application Integration: Extending the system with a mobile application to allow care seekers and caretakers to access services remotely.
3. Real-Time Notifications: Adding SMS, email, or in-app notifications for care request updates, assignments, and status changes. Enhanced

4. Online Payment Gateway: Integrating secure payment gateways to support real-time transactions and billing management.
5. Security & Data Privacy: Enhancing system security with data encryption, secure authentication, and role-based access control.
6. Scalability & Cloud Deployment: Deploying the system on cloud infrastructure to support multiple users, large datasets, and institutional-level usage.

These enhancements will improve the system's efficiency, reliability, and scalability, making Care Connect more suitable for real-world caretaker management applications and future technological expansion.

## VI. CONCLUSION

The proposed Care Connect – Desktop Caretaker Management System successfully integrates caretaker registration, care request handling, caretaker assignment, and service tracking into a unified desktop-based platform. The implementation using Electron.js, HTML, CSS, JavaScript, and Node.js demonstrates efficient role-based access, structured workflow management, and reliable system operation. The simulation results confirm that the system performs consistently and provides accurate care status updates across different user roles. By offering a centralized and user-friendly interface, the system reduces manual coordination, improves transparency, and enhances service efficiency. The Care Connect application serves as a practical and effective solution for academic projects and small-scale caretaker management environments.

## ACKNOWLEDGMENT

I would like to express my sincere gratitude to Sau. Sundarbai Manik Adsul Polytechnic, Chas, Ahilyanagar, Maharashtra, for providing the necessary guidance, infrastructure, and academic support to successfully complete this project. I am thankful to my project guide and faculty members for their valuable suggestions, continuous encouragement, and technical assistance throughout the development of the Care Connect – Desktop Caretaker Management System.

I also extend my appreciation to the developers and open-source communities behind Electron.js, Node.js, HTML, CSS, and JavaScript, whose tools and documentation played a crucial role in the implementation of this project. Finally, I would like to acknowledge the support and motivation provided by my family and friends, which greatly contributed to the successful completion of this work.

## REFERENCES

[1] Electron.js Documentation. Available: https://www.electronjs.org/docs
[2] Node.js Official Documentation. Available: https://nodejs.org/en/docs
[3] HTML, CSS, and JavaScript Web Standards. Available: https://developer.mozilla.org
[4] SQLite Database Documentation. Available: https://www.sqlite.org/docs.html
[5] PostgreSQL Official Documentation. Available: https://www.postgresql.org/docs
[6] A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, 6th ed., McGraw-Hill, 2011.
[7] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed., McGraw-Hill, 2010.
[8] I. Sommerville, Software Engineering, 10th ed., Pearson Education, 2016.
[9] J. Sommerville and P. Sawyer, "Requirements engineering: A good practice guide," Wiley, 1997.
[10] A. Dennis, B. Wixom, and D. Tegarden, Systems Analysis and Design: An Object-Oriented Approach, Wiley, 2015.
[11] S. McConnell, Code Complete, 2nd ed., Microsoft Press, 2004.
[12] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modelling Language, 3rd ed., Addison-Wesley, 2003.
[13] IEEE Std 830-1998, "IEEE Recommended Practice for Software Requirements Specifications," IEEE, 1998.
[14] P. Laplante, Requirements Engineering for Software and Systems, CRC Press, 2017.
[15] N. Wirth, "Program development by stepwise refinement," Communications of the ACM, vol. 14, no. 4, pp. 221–227, 1971.

[16] J. Nielsen, “Usability engineering for interactive systems,” Morgan Kaufmann, 1993.
[17] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2001.
[18] GitHub Opensource Community Documentation. Available: https://docs.github.com