XZ-1 - Development of a Dementia Care Voice Assistant application utilizing Large Language Models (LLMs)

CS 4850 - Fall 2024

December 2024

Sharon Perry

Emily Centeno

Ronald Bates

Project Portfolio Webpage
Project Repository

Lines of Code: 1237
Project Components:50
Total Hours: 195

Table of Contents

Introduction

Our aim is to develop a voice-assisted application that can offer critical support to informal caregivers of individuals with Alzheimer's disease and related dementias (ADRD) through seamless and intuitive voice interactions. Recognizing the challenges faced by caregivers, such as stress, limited access to real-time information, and the need for practical solutions to manage challenging behaviors, this application will be designed as a supportive companion. By leveraging advanced technology, it will help to alleviate some of these burdens and empower caregivers with reliable, accessible information and guidance right when they need it.

At the core of the app is a fine-tuned Retrieval-Augmented Generation (RAG) based Large Language Model (LLM). This model is customized specifically to deliver contextually accurate and relevant responses to caregivers' inquiries, ensuring that the guidance provided is immediate and personalized to their unique caregiving situations. The use of a RAG-based LLM allows the app to pull from a variety of up-to-date, evidence-based sources, presenting information that aligns with best practices in dementia care and is directly applicable to common caregiving challenges. This means that caregivers can ask questions about managing symptoms, responding to behavioral issues, or addressing emotional well-being, and the app will provide detailed, actionable advice instantly.

In terms of user experience, our focus is on developing a user-friendly interface capable of understanding natural voice commands, simplifying navigation, and reducing the need for complex interactions that might overwhelm users. The app will be tailored to allow caregivers to speak naturally, receiving clear, concise, and compassionate responses. Additionally, this voice-interactive interface is designed to offer an immersive and engaging experience, making it easy for caregivers to integrate the app into their daily routines without interrupting the flow of caregiving tasks.

The envisioned outcome is a fully functional iOS app that provides caregivers with instant, non-pharmacological, evidence-based interventions for behavioral symptom management in dementia care. These interventions are designed to be supportive and respectful of both the caregiver's and the care recipient's needs, offering practical solutions that are backed by current research in dementia care. By focusing on non-pharmacological interventions, we aim to offer caregivers tools to manage symptoms in a way that reduces reliance on medications and focuses instead on effective behavioral strategies.

Ultimately, this application aspires to be a comprehensive support system that empowers caregivers to manage the emotional and behavioral symptoms of dementia with confidence and ease. The app's personalized responses and voice-command capabilities make it a unique tool in the landscape of caregiver support, promoting a more manageable, compassionate caregiving experience. Through this app, we hope to improve the quality of life for both caregivers and individuals with ADRD by delivering information that is both practical and rooted in empathy, making caregiving a more supported and informed journey.

Collaboration and Communication

Communication: Teams and Discord

Collaboration: Teams and Discord

Version Control: GitHub

Requirements

1. Enables Multiple Caregivers to Access Shared Patient Data and Chat History

- **Priority Level**: High
- **Level of Effort**: High
- **Acceptance Criteria**:
    - Multiple caregivers can access a shared patient profile.
    - All authorized caregivers can view patient data and chat history.
    - Changes made by any caregiver are visible in real-time to others.
    - Data access complies with privacy and security regulations (e.g., HIPAA).

2. Input Validation

- **Priority Level**: High
- **Level of Effort**: Medium
- **Acceptance Criteria**:
    - The system checks all user inputs for correctness, ensuring required fields are completed.
    - Invalid inputs prompt user-friendly error messages.
    - Data entry fields validate format requirements (e.g., email format, phone numbers).
    - Malicious inputs (e.g., SQL injections, XSS) are prevented.

3. Create Account

- **Priority Level**: High
- **Level of Effort**: Medium
- **Acceptance Criteria**:
    - New users can register by entering required information (e.g., name, email, password).
    - Account creation includes password strength validation and a “Terms of Service” acceptance checkbox.
    - Users receive an email verification link, and only verified accounts can access the app.

4. Evidence-Based Responses

- **Priority Level**: High
- **Level of Effort**: High
- **Acceptance Criteria**:
    - The app provides responses based on current, evidence-based behavioral intervention research.
    - Responses are clearly referenced, where applicable, to reassure caregivers of their validity.
    - Feedback shows that users consistently receive relevant and accurate responses for caregiving challenges.

5. Contextual Assistance

- **Priority Level**: Medium
- **Level of Effort**: High
- **Acceptance Criteria**:
    - The app provides responses based on prior interactions or stored patient data to give contextually relevant advice.
    - Users experience seamless transitions between topics without repetitive or redundant responses.
    - Assistance is personalized based on user history and patient data, if available.

6. User Authentication

- **Priority Level**: High
- **Level of Effort**: Medium
- **Acceptance Criteria**:
    - Users must log in with a valid username and password to access the app.
    - The system provides secure authentication protocols (e.g., multi-factor authentication).
    - Passwords are encrypted, and user sessions are secured to prevent unauthorized access.

7. Intuitive User Interface

- **Priority Level**: High
- **Level of Effort**: Medium
- **Acceptance Criteria**:
    - The interface design is easy to navigate, with a focus on simplicity for non-technical users.
    - Caregivers can quickly access key features without confusion or excessive steps.
    - Interface usability testing shows high satisfaction scores from target user groups.

8. Patient Profile

- **Priority Level**: High
- **Level of Effort**: Medium
- **Acceptance Criteria**:
    - Caregivers can create and edit a patient profile, including fields for medical history, medications, and behavior notes.
    - The patient profile is accessible to all authorized caregivers and securely stored.
    - Profile updates synchronize across caregiver accounts in real-time.

9. Voice-to-Text Functionality

- **Priority Level**: Medium
- **Level of Effort**: High
- **Acceptance Criteria**:
    - The app accurately transcribes voice inputs from caregivers.
    - Transcription captures both short and long phrases without significant errors.
    - Users find the voice-to-text feature reliable and responsive to natural speech.

10. Non-Pharmacological Guidance

- **Priority Level**: High
- **Level of Effort**: High
- **Acceptance Criteria**:
    - The app offers a range of non-pharmacological interventions based on specific behaviors or symptoms.
    - Interventions are evidence-based, focusing on alternative approaches rather than medications.
    - Caregivers can request specific guidance and receive relevant, applicable responses promptly.

11. 911 Functionality

- **Priority Level**: Medium
- **Level of Effort**: High
- **Acceptance Criteria**:
    - Users can activate an emergency call to 911 directly from within the app if needed.
    - The app requires user confirmation before dialing emergency services.
    - Emergency functionality meets local regulations and includes a safety mechanism to prevent accidental calls.

Analysis

Focus Group

The primary users of the voice assistant app are informal caregivers of people with Alzheimer's Disease and related dementia (ADRD). These caregivers may include family members, friends, or hired professionals who assist patients with day-to-day activities. The target audience is typically non-technical and may not have significant experience with technology, which means the app must be simple, intuitive, and efficient. Caregivers are often under stress and may require immediate, easy-to-access support to manage the complex needs of patients with dementia.

Key Needs of the End User:

- **Immediate, context-specific support**: Caregivers often need rapid, relevant information on handling behavioral symptoms of patients (e.g., aggression, anxiety, confusion).
- **Ease of Use**: The app should have a user-friendly interface that requires minimal technical expertise.
- **Voice Interaction**: Given the potentially limited mobility or cognitive impairment of patients and caregivers, a voice assistant that responds to voice commands will be essential for providing hands-free assistance.
- **Real-time Intervention**: The app should provide quick, evidence-based interventions that can be immediately applied to calm or redirect a patient.

System Requirements

The system will be built to meet both functional and non-functional requirements:

Functional Requirements:

- **Voice Command Processing**: The app must accurately recognize voice inputs from caregivers and provide appropriate responses. This includes symptom-related queries, requests for help, or checking patient information.
- **Contextual Responses via LLM**: The app must leverage the fine-tuned Retrieval-Augmented Generation (RAG) model to deliver accurate, personalized responses based on caregiver queries.
- **User Authentication**: The app must securely authenticate caregivers through Firebase, ensuring that patient data remains confidential and accessible only to authorized individuals.
- **Non-pharmacological Interventions**: The app should suggest strategies based on current evidence to help manage behavioral symptoms without resorting to medication.
- **Future Features**: Planned future features include:
    - **911 Detection**: Recognizing emergency situations and alerting authorities.
    - **Group Profiles**: Enabling multiple family members or caregivers to access and manage a shared patient profile.

Non-Functional Requirements:

- **Usability**: The app must be intuitive, especially for non-technical users. This involves simple navigation and clear instructions.

- **Performance**: The system should be responsive, with minimal delay in voice recognition and response generation.
- **Scalability**: The backend, especially the LLM, should be able to scale as new features are added or more users join.
- **Security**: Sensitive data, including patient information, must be encrypted during transmission and stored securely in compliance with privacy regulations (such as HIPAA).

Technical Requirements
The app will rely on the following key technologies:

- **iOS Development**: The app will be developed for iOS using Swift and Xcode.
- **Firebase**: Firebase will handle authentication (e.g., email and password login) and data storage for user profiles and patient records.
- **LMStudio**: The app will use LMStudio for managing the fine-tuned RAG-based LLM, enabling dynamic responses based on user queries.
- **Voice Recognition**: Speech-to-text capabilities will be integrated to allow caregivers to interact with the app using voice commands.
- **Material Design**: The user interface will follow Material Design principles to ensure accessibility and a seamless experience for caregivers.

Use Cases
Several use cases outline the interaction between the caregiver and the app:

- **Use Case 1: Caregiver Logs In**
    - **Description**: The caregiver authenticates using email and password.
    - **System Behavior**: The app validates credentials via Firebase and grants access to the dashboard.
- **Use Case 2: Caregiver Asks for Intervention Suggestion**
    - **Description**: The caregiver asks for a non-pharmacological intervention for a specific behavioral symptom (e.g., “What can I do if my patient is agitated?”).
    - **System Behavior**: The app uses the LLM to search for relevant, evidence-based interventions and provides a list of strategies or techniques.
- **Use Case 3: Caregiver Logs Patient’s Symptoms**
    - **Description**: The caregiver records observations about the patient’s condition, such as mood or behavior changes.
    - **System Behavior**: The app stores the data securely in Firebase and provides reminders to log observations regularly.
- **Use Case 4: Caregiver Requests Help in Emergency**
    - **Description**: In case of an emergency, the caregiver can request urgent help, including alerting emergency services.
    - **System Behavior**: The app processes the emergency request and triggers a 911 call (future feature) or sends alerts to family members.
- **Use Case 5: Caregiver Queries Patient’s Medication Schedule**
    - **Description**: The caregiver asks about the patient’s medication routine.
    - **System Behavior**: The app retrieves and displays the patient’s medication schedule, including dosage and timing.

Assumptions and Dependencies

- **Voice Recognition Accuracy**: The success of the app's voice recognition depends on clear audio input. Background noise or unclear speech may result in incorrect responses.
- **Availability of LLM Responses**: The app's effectiveness in providing meaningful answers relies on the accuracy and relevance of the fine-tuned RAG-based LLM.
- **User Adoption**: Caregivers may need some time to trust and integrate the app into their routine, especially if they are unfamiliar with technology.

Design

Software Dependencies:

- **LMStudio**: Required for managing the fine-tuned Language Model (LLM) and handling API calls. It must be compatible with the current version of the LLM and support the necessary integrations.
- **Swift**: The app is developed using Swift, which necessitates Xcode for development and testing. The latest stable version of Xcode should be used to ensure compatibility with Swift and iOS features.
- **Material Design Components**: Used for the UI elements, which may require additional libraries or frameworks compatible with Swift to ensure proper implementation of Material Design principles.

Hardware Dependencies:

- **iOS Devices**: The application will be designed to run on iPhones. Testing should be conducted on a range of iOS devices to ensure compatibility and performance.
- **Microphone**: The app relies on the device's microphone for voice commands. Users must have a functioning microphone, and hardware variations should be accounted for during testing.

End-User Characteristics:

- **Caregivers**: The primary users are informal caregivers of individuals with Alzheimer's Disease and Related Dementias (ADRD). They may not be tech-savvy and will benefit from a user-friendly and intuitive interface. Accessibility features should be incorporated to assist users with varying levels of tech proficiency and potential disabilities.
- **Multilingual Support**: Some users may prefer or require the application in languages other than English. The app should support multiple languages to accommodate a diverse user base.

Modular Architecture:

- **Component-Based Design**: Utilized a modular architecture to break down the application into manageable components such as the voice command interface, symptom guidance, and multimedia responses. This allows for easier development, testing, and maintenance.
- **Separation of Concerns**: Ensured that different aspects of the app, such as the user interface, backend services, and LLM integration, are clearly separated to enhance scalability and flexibility.

Compliance:

- **Security and Privacy**: Adhered to relevant data protection regulations (e.g., HIPAA) to ensure the secure handling of user information. Implemented best practices for data encryption and user authentication.
- **Performance Optimization**: Focused on optimizing app performance to ensure quick load times and responsive interactions, enhancing the overall user experience.

UI Design Framework:

• **Decision**: The app will use Material Design principles for UI components.

• **Reasoning**: Material Design offers a consistent and modern look that enhances usability and accessibility. It provides pre-designed components that ensure a visually appealing and user-friendly interface. This aligns with the goal of creating an intuitive experience for caregivers, who may not be tech-savvy.

• **Trade-offs**: Implementing Material Design in Swift may require additional libraries or custom components, which could increase initial development effort.

Integration of LLM with LMStudio

• **Decision**: The pre-existing Language Model (LLM) will be integrated using LMStudio for managing API calls and processing user inputs.

• **Reasoning**: LMStudio provides a robust environment for managing the LLM and handling complex natural language processing tasks. This integration ensures that the app can leverage advanced language capabilities without reinventing the wheel. It aligns with the goal of providing accurate and contextually relevant responses to caregivers.

• **Trade-offs**: Reliance on LMStudio introduces a dependency on external tools, which may require updates or adjustments if the tool evolves or if compatibility issues arise.

Modular Architecture

• **Decision**: The application will be organized into modular components, such as the voice command interface, symptom guidance, and multimedia responses.

• **Reasoning**: A modular architecture facilitates easier development, testing, and maintenance. Each module can be developed and updated independently, allowing for scalability and flexibility. This design supports the goal of iterative improvement and agile development.

• **Trade-offs**: Modularization can introduce complexity in terms of managing intermodular communication and ensuring seamless integration. Careful planning and documentation is required to mitigate these challenges.

Scalability and Performance

• **Decision**: The app will be designed to handle multiple concurrent users with optimal performance.

• **Reasoning**: Designing for scalability ensures that the app can accommodate growth and high user activity without performance degradation. This supports the goal of providing a reliable service to a potentially large user base.

• **Trade-offs**: Ensuring scalability may require additional architectural considerations and infrastructure investments. The focus on performance must be balanced with cost and resource constraints.

System Architecture

The system is designed with a modular architecture to enhance manageability, scalability, and maintainability. The overall functionality is divided into several high-level subsystems or components, each responsible for specific aspects of the application. This partitioning allows for clear separation of

concerns, efficient development, and integration of features, ultimately leading to a cohesive and user-friendly application.

Key Subsystems and Components

User Interface (UI) Component

• **Responsibilities**: This component handles all user interactions, including the presentation of information, navigation, and user input. It is built using Material Design principles to ensure a consistent and intuitive experience.

• **Functionality**: Provides the primary interface through which users interact with the app, including voice command input, text display, multimedia responses, and navigation between different sections of the app.

• **Integration**: Communicates with the Voice Command Processing component to handle user inputs and display relevant responses.

Voice Command Processing Component

• **Responsibilities**: Manages the recognition and processing of spoken commands and queries. Utilizes the Apple Speech framework for converting speech to text and interacting with the Language Model (LLM).

• **Functionality**: Converts spoken input into text, interprets the intent, and routes the information to the appropriate subsystem or retrieves relevant responses from the LLM.

• **Integration**: Interfaces with both the UI Component to handle voice interactions and the LLM Management component to process and respond to queries.

Detailed System Design

Functional Requirements

1. **User Feedback Collection (Functional)**
   Success Standard: The system collects user feedback via prompts and voice responses, with feedback incorporated into system improvements

2. **Multimedia Responses (Functional)**
   Success Standard: The system provides responses in text and audio formats.

3. **Emergency Recognition (Functional)**
   Success Standard: The system can detect emergency situations and alert caregivers or emergency services.

4. **Support for Multiple Caregivers (Functional)**
   Success Standard: Multiple caregivers can access and manage a patient’s profile.

5. **Predictive Capabilities (Functional)**
   Success Standard: The system analyzes interactions to predict and address potential questions or concerns.

6. **Localization (Functional)**
   Success Standard: The system supports multiple languages and cultural nuances.

7. **Voice Command Interface (Functional)**
   Success Standard: Allows caregivers to interact with the app through voice commands. Provides instant, spoken responses and guidance.

8. **Symptom Management (Functional)**
   Success Standard: Offers both text and voice feedback that enables the caregivers to search for specific symptoms and receive evidence-based strategies.

Nonfunctional Requirements

1. **App Performance (Nonfunctional)**
   Success Standard: The app loads within 3seconds and responds to user inputs within 1 second.

2. **User Interface (Nonfunctional)**
   Success Standard: The app has a simple, intuitive, and accessible interface for elderly users.

3. **Scalability (Nonfunctional)**
   Success Standard: The app supports multiple concurrent users without performance degradation.

4. **Security Compliance (Nonfunctional)**
   Success Standard: The app complies with relevant healthcare data protection regulations

Development

Outline for Voice Assistant Application for Dementia Patients

1. **User-Centric Design for Dementia Care**
    a. Focus on ease of use and accessibility.
2. **Contextual Response System**
    a. Real-time, evidence-based responses tailored to caregivers' needs.
    b. RAG-based language model (LLM) fine-tuned for dementia care scenarios.
3. **Voice Command Integration**
    a. Voice-to-text capabilities for intuitive caregiver interaction.
    b. 911 detection functionality for emergencies.
4. **User Authentication and Security**
    a. Secured access for caregivers with user authentication.
    b. Emphasis on data protection for sensitive patient information.
5. **Collaborative Family Profiles**
    a. Group profiles enabling family members to stay informed and connected.
    b. Shared access for caregivers in a patient's network.
6. **Non-Pharmacological Behavioral Intervention Information**
    a. Immediate access to non-pharmacological intervention methods.
    b. Guidance for managing common dementia-related behavioral symptoms.
7. **Implementation and Development Tools**
    a. Figma and Firebase for UI/UX and backend services.
    b. Swift, LMStudio, and Xcode for developing and testing the iOS application.

Concept Details and Implementation

1. **User-Centric Design for Dementia Care**
    a. *Implementation*: The app's UI is developed in Figma, focusing on simplified icons, large text, and a clean layout to minimize cognitive load for users with dementia. Each screen limits information density, using intuitive navigation to avoid confusion. Accessibility features, like voice guidance and adjustable font sizes, are integrated to enhance usability.

2. **Contextual Response System**
    a. *Implementation*: The application utilizes a fine-tuned Retrieval-Augmented Generation (RAG) LLM to ensure responses are contextually relevant to dementia caregiving. This LLM, managed via LMStudio, can process caregiver queries about dementia symptoms and offer responses backed by behavioral intervention techniques. The model is trained on dementia-specific datasets to address common scenarios caregivers face, enabling swift, personalized assistance.

3. **Voice Command Integration**
    a. *Implementation*: Voice-to-text functionality is being integrated to capture caregiver queries seamlessly. Using speech recognition libraries in Swift, the app processes voice input and converts it into actionable text for the LLM to analyze. In addition, an emergency detection feature (911 detection) is planned to monitor for phrases like “help” or “emergency,” alerting emergency contacts automatically if needed.

4. **User Authentication and Security**
    a. *Implementation*: User authentication is built using Firebase to protect sensitive data. Multi-factor authentication (MFA) enables us to add an extra layer of security. Security protocols include encryption for data storage and transmission, ensuring compliance with HIPAA and other health data regulations. This setup secures the private health information shared by caregivers about the dementia patients they care for.

5. **Collaborative Family Profiles**
    a. *Implementation*: Firebase is used to manage group profiles for families connected to a patient. Each profile can include multiple caregivers, allowing family members to view patient status updates and contribute relevant notes. This setup fosters collaboration and shared responsibility among family members, with individual permission based on roles.

6. **Non-Pharmacological Behavioral Intervention Information**
    a. *Implementation*: The app provides non-pharmacological intervention suggestions drawn from evidence-based databases, offering immediate guidance on common dementia behaviors like agitation or wandering. The RAG model retrieves targeted responses from a curated knowledge base to ensure the information is accurate and practical for caregivers in real-time scenarios.

7. **Implementation and Development Tools**
    - *Implementation*: The app leverages Figma for prototype design and user flow mapping, Firebase for backend support and user management, and Xcode with Swift for iOS development. LMStudio serves as the platform for deploying and managing the LLM, allowing seamless API integration. This stack enables a robust and responsive user experience, from the UI to the backend AI interactions.

Detailed Steps with Implementation Notes

1. **Designing Mockups in Figma**
    a. *Details*: Created interactive wireframes of the user interface, focusing on the chat screen, settings, login, and the main dashboard.
    b. *Setup*: Ensure that the UI/UX flow aligns with the caregiver's needs by conducting reviews with a few caregivers or care industry experts, if possible.
    c. *Implementation*: Exported assets and annotated screens, then linked relevant design components to help with the development.
2. **Setting up the Project in Xcode**
    a. *Setup*: Opened Xcode, created a new iOS project, and chose Swift as the programming language.
    b. *Implementation*: Configured the project settings, including bundle ID and deployment targets, and integrated Material components for the UI elements.
3. **Creating a Git Repository**
    a. *Setup*: Initialized a local Git repository, pushed it to GitHub (or another version control platform), and set up branch protections for stable development.
    b. *Implementation*: Established a development branch, with separate branches for different features.

4. **Downloading and Setting Up the LLM in LMStudio**
    a. *Setup*: Accessed LMStudio, downloaded the fine-tuned RAG model, and configured it as the primary source for generating responses.
    b. *Implementation*: Connected the LLM to the project via LMStudio's API, verified it was accessible and responsive, and tested it with sample data.
5. **Creating an API Client for App-LLM Communication**
    a. *Setup*: Built an API client in Swift to manage requests and responses from the LLM.
    b. *Implementation*: Designed the client with error handling and timeouts for network resilience and created reusable functions for API calls.
6. **Implementing Voice-to-Text Functionality**
    a. *Setup*: Integrated iOS Speech Framework to handle voice-to-text processing.
    b. *Implementation*: Added permissions for microphone access, set up real-time transcriptions, and ensured accuracy by testing various phrases related to caregiving.
7. **Setting Up Chat UI for API Integration**
    a. *Setup*: Created a chat interface in Swift, designed to handle user input and response display in real-time.
    b. *Implementation*: Applied custom styles to differentiate between user and system messages and utilized Material UI components for a clean, intuitive interface.
8. **Sending Voice-to-Text Data to the API**
    a. *Setup*: Modified the chat screen's input process to convert speech-to-text results directly into API requests.
    b. *Implementation*: Integrated a function that captures and sends transcribed text to the LLM API, providing users with quick responses.
9. **Enabling Phone Calls (Without 911 Detection)**
    a. *Setup*: Implemented a "Call" button within the app using iOS's *UIApplication.shared.open()* method.
    b. *Implementation*: Added logic to prompt users with a call confirmation, making it user-initiated until automated detection is ready.
10. **Building a Firebase App for User Authentication and Chat History**
    a. *Setup*: Set up Firebase, configured authentication methods, and established a Realtime Database or Firestore for chat history.
    b. *Implementation*: Linked user accounts with chat sessions to save each user's unique history and preferences.
11. **Implementing Firebase UI for Login Management**
    a. *Setup*: Integrated Firebase UI to simplify login, enabling social logins if needed, and streamlined user management.
    b. *Implementation*: Configured Firebase UI to prompt users with custom or pre-built login screens, adding options to handle forgotten passwords.
12. **Updating Chat Functions to Store API Data in Firebase**
    a. *Setup*: Refactored the chat functions to save both outgoing API requests and incoming responses in Firebase.
    b. *Implementation*: Ensured data privacy and storage optimization by structuring the database to handle user-session data securely.
13. **Integrating Aaron's RAG Model for Fine-Tuned Responses**

a. *Setup*: Connected the RAG model with LMStudio, adjusted the API client to query the updated model, and ran sample tests to evaluate improvements.
b. *Implementation*: Tuned response parameters to enhance relevance and accuracy in caregiver-related responses, providing a more personalized support experience.

14. **UI Cleanup and Settings Page Development**
    a. *Setup*: Conducted a UI audit to standardize design elements across screens.
    b. *Implementation*: Added a settings page where users can edit personal details, reset passwords, and manage notification preferences.
15. **Unit Testing Throughout Steps 5-13**
    a. *Implementation*: Implemented unit tests to validate API responses, voice-to-text accuracy, Firebase data syncing, and RAG model functionality, iteratively resolving issues for each core feature.

Testing

Requirement	Description	Test Steps	Expected Results	Pass/Fail	Severity	Actual Result	Comments
Enables multiple caregivers to access shared patient data and chat history.	Enables multiple caregivers to access shared patient data and chat history.	Add a second caregiver and confirm shared access to patient data.	All caregivers can access and view shared patient data	Pass	High	Successful	N/A
Input validation	Ensures data entered is correct and complete before submission.	Enter invalid data and confirm app shows error message.	App displays error for invalid entries and prevents submission.	Fail	High	Unsuccessful	N/A
Create Account	Allows new users to register securely with email or social logins.	Attempt new user registration and confirm account creation.	New account is created, and user is redirected to the home screen	Pass	Mid	Successful	N/A
Evidence Based Responses	Provides responses based on verified, non-pharmacological interventions.	Ask a sample question and verify response is evidence-based.	Response provided is accurate and based on clinical guidelines	Pass	High	Successful	N/A
Contextual Assistance	Offers personalized support based on caregiver queries and patient needs	Ask patient-specific questions and check for tailored response.	Response is relevant to user's question and patient's profile.	Pass	Mid	Successful	N/A
User Authentication	Secures app access with Firebase-powered login and identity verification.	Log in with valid credentials and confirm access granted.	User is successfully logged in with valid credentials.	Pass	High	Successful	N/A
Intuitive User Interface	Simple, easy-to-navigate interface tailored for quick access to features.	Navigate all screens and verify ease of use and accessibility.	All screens are accessible and easy to navigate without errors.	Pass	High	Successful	N/A
Patient Profile	Stores details about the patient for tailored support and caregiver insights.	Update patient info and verify details are saved correctly.	Patient details are updated and stored accurately.	Pass	High	Successful	N/A
Voice-to-Text Functionality	Converts spoken caregiver input into text for seamless interaction.	Speak a command and confirm accurate text conversion.	Spoken input is accurately converted into text.	Pass	High	Successful	N/A
Non-Pharmacological Guidance	Suggests alternative behavioral strategies for managing symptoms.	Request symptoms help and verify non-drug advice is provided.	App provides appropriate, non-drug-related advice.	Pass	Mid	Successful	N/A

911 Functionality	Ensure emergency detection functions properly	Attempt to call 911	Automatically prompts emergency actions	Pass	High	Successful	N/A

Version Control

Tool Used:

- Git for version control, hosted on GitHub.

Repository Information:

- URL: https://github.com/Dementia-Voice-Care-Assistant
- Repository Name: Dementia Voice Care Assistant
- Access: Public

Branching Strategy:

- Main Branch: The main branch was used for stable, production-ready code.

Commit Practices:

- Commit Messages: Followed clear, descriptive messages using the [chosen convention, e.g., Conventional Commits or custom].

Collaboration and Pull Requests:

- All feature development and bug fixes were merged into the main branch via pull requests.
- Pull requests included code reviews by team members and checks for conflicts.

Backup and Recovery:

- GitHub served as a central backup for all project-related files.
- Regular snapshots of the repository ensured safe recovery in case of data loss.

Summary

The project involved developing a voice-assisted iOS application to support informal caregivers of individuals with Alzheimer's disease and related dementias (ADRD). Designed to address the challenges of dementia caregiving, the app provided timely, evidence-based support through a virtual assistant capable of delivering non-pharmacological intervention information. Its core features included chatbot functionality for responding to caregiver queries, secure user authentication, and plans for future enhancements such as voice-to-text, 911 emergency detection, and family group profiles. Powered by a fine-tuned Retrieval-Augmented Generation (RAG) Language Model (LLM), the app ensured reliable, context-sensitive responses. The development utilized Swift and Material for the frontend, Firebase for backend data management and authentication, and LMStudio for managing the LLM and API calls. Built with an agile approach, the project incorporated regular testing and feedback, leveraging Figma for UI/UX design and Xcode for app development and debugging.

Appendix

UI Mock-Ups