|  |
| --- |
|  |
|  |
| **AI Wellness Companion App** |
| for Aging in Place |

|  |
| --- |
| **Course**: CSIS 4495 – Applied Research Project  **Term**: Fall 2025  **Section**: 050  **Team Members:**  - Aafrin Zahid Memon (300388614) - Team Lead  - Akinro Akintunde(300389708) - Member  - Ifeoluwa Aribo(300389564) – Member  **Date**: 23rd October 2025 |

Table of Contents

[1 Introduction 2](#_Toc151721313)

[2 Proposed Research Project 2](#_Toc151721314)

[3 Project Planning and Timeline 2](#_Toc151721315)

[4 Implemented Feature 2](#_Toc151721316)

[Overview 2](#_Toc151721317)

4.2 [Details of Implementation 1: Authentication (Single Screen Login / Sign Up) 2](#_Toc151721318)

[4.2 Details of Implementation 1(Monitoring and Tracking) 2](#_Toc151721318)

[4.3 Details of Implementation: Ai Powered Chat and voice Feature 2](#_Toc151721319)

## [4.4 Details of Implementation: Cognitive Improvement Games…………………………………………….2](#_Toc151721318)

[5 Lessons Learned and Future Work 3](#_Toc151721320)

[6 Concluding Remarks 3](#_Toc151721321)

[7 Appendix 3](#_Toc151721322)

[7.1 Appendix A: Installation Guide 3](#_Toc151721323)

[7.2 Appendix B: User Guide 3](#_Toc151721324)

[7.3 Appendix C: Dataset and API Used 3](#_Toc151721325)

[7.4 Appendix D: Hardware, Software, Cloud, Architecture, etc 3](#_Toc151721326)

[7.5 Appendix E: Code Explanation 3](#_Toc151721327)

# Introduction

This midterm report presents the development progress of an AI-powered wellness companion mobile application designed specifically for aging populations. The project aims to address the growing need for accessible, user-friendly health monitoring and cognitive support tools for elderly users. The application leverages modern mobile technologies including React Native, Expo, and TypeScript to create a comprehensive platform that combines health monitoring, medication management, cognitive exercises, and AI-powered voice assistance. The target demographic consists of seniors aged 65 and above who require assistance with daily health management and cognitive stimulation. The development team has focused on creating an intuitive user interface with large, clear buttons, voice-activated features, and simplified navigation to accommodate users with varying levels of technical proficiency. The application addresses critical needs in elderly care including medication adherence, physical activity tracking, sleep monitoring, and cognitive health maintenance.

# Proposed Research Project

The Health Monitoring System serves as the cornerstone of the application, providing users with comprehensive tools to track and manage their physical well-being. The breathing exercises component incorporates guided meditation techniques specifically designed for elderly users, featuring visual cues and audio guidance that help users maintain proper breathing patterns. These exercises are not merely recreational activities but therapeutic interventions that can help reduce stress, improve cardiovascular health, and enhance overall respiratory function. The implementation includes multiple breathing patterns such as the 4-7-8 technique, box breathing, and deep breathing exercises, each with customizable durations to accommodate different fitness levels and preferences.

Sleep cycle tracking and analysis represents another critical component of the health monitoring system. This feature allows users to log their sleep patterns, including bedtime, wake time, and sleep quality assessments. The system analyzes this data to provide insights into sleep patterns, identify potential sleep disturbances, and offer personalized recommendations for improving sleep hygiene. The interface is designed to be simple and intuitive, requiring minimal input while providing maximum value through automated analysis and trend identification.

Physical activity monitoring encompasses walking and stretching exercises, providing users with structured activities that promote mobility and cardiovascular health. The walking tracker component includes features such as step counting, distance measurement, and pace monitoring, while the stretching module offers guided exercises specifically designed for elderly users, taking into account common mobility limitations and safety considerations. These features work together to encourage regular physical activity while ensuring user safety and providing appropriate modifications for different fitness levels.

The vital signs tracking interface provides a centralized location for users to record and monitor key health indicators such as blood pressure, heart rate, weight, and other relevant metrics. This feature enables users to maintain comprehensive health records and identify trends or patterns that may require medical attention. The interface is designed with accessibility in mind, featuring large buttons, clear labels, and simple data entry methods that accommodate users with varying levels of technical proficiency.

The AI-Powered Voice Assistant represents a sophisticated integration of artificial intelligence capabilities designed to provide intelligent health support and assistance. The natural language processing component enables users to interact with the application using conversational language, making the technology more accessible and intuitive.

 Users can ask questions about their health, request information about medications, or seek guidance on health-related topics using natural speech patterns rather than navigating complex menu systems.Voice-activated medication reminders represent a critical safety feature that helps ensure medication adherence, a common challenge among elderly users.

The Medication Management system addresses one of the most critical aspects of elderly health care: ensuring proper medication adherence and safety. The automated reminder system provides timely notifications for medication doses, taking into account individual schedules, time zones, and personal preferences.

Dosage tracking functionality enables users to record medication intake, providing a comprehensive record that can be shared with healthcare providers. This feature helps users maintain accurate medication logs while providing healthcare professionals with valuable information about medication adherence patterns. The interface is designed to be simple and quick to use, encouraging regular logging without creating additional burden.

**Research Objectives:**

* Develop a user-friendly mobile application that addresses the unique health and wellness needs of aging populations
* Implement AI-powered voice assistance to provide 24/7 health monitoring and support
* Create an integrated platform combining physical health tracking, medication management, and cognitive exercises
* Evaluate the effectiveness of mobile health interventions in improving quality of life for elderly users

**Technical Architecture:**

* **Frontend:** React Native with Expo framework
* **Language:** TypeScript for type safety
* **UI Framework:** Custom components with LinearGradient styling
* **Navigation:** React Navigation for seamless screen transitions
* **State Management:** React Hooks for local state management
* **Platform Support:** Cross-platform (iOS, Android, Web)

# Project Planning and Timeline

**Phase 1: Project Initiation and Planning (Weeks 1-2)**

* Project requirements analysis
* Technology stack selection
* UI/UX design planning
* Development environment setup

**Phase 2: Core Development (Weeks 3-6)**

* Basic application structure implementation
* Screen navigation system development
* Core health monitoring features
* Voice chat interface implementation
* Cognitive games development

**Phase 3: Feature Integration (Weeks 7-8)**

* AI voice assistant integration
* Medication reminder system
* Data persistence implementation
* Cross-platform testing

**Phase 4: Testing and Optimization (Weeks 9-10)**

* User interface refinement
* Performance optimization
* Accessibility improvements
* Bug fixes and stability enhancements

**Phase 5: Deployment and Documentation (Weeks 11-12)**

* Final testing and quality assurance
* User documentation creation
* Deployment preparation
* Final report submission

**Current Status (Midterm):**

* **Completed:** 65% of core functionality
* **In Progress:** UI/UX refinements and testing of Frontend and Implementation phase of Backend Features
* **Remaining:** Final integration, testing, and deployment

**Key Deliverables Completed:**

* Functional mobile application with 8+ screen components
* Voice chat interface with AI simulation
* Health monitoring dashboard
* Cognitive exercise modules
* Medication management system
* Cross-platform compatibility (iOS, Android)

**Technical Achievements:**

* Successfully implemented React Native with Expo framework
* Created responsive UI components with accessibility features
* Integrated voice recognition and text-to-speech capabilities
* Developed modular screen architecture for easy maintenance
* Implemented real-time data tracking and visualization

This midterm report demonstrates significant progress toward creating a comprehensive AI wellness companion that addresses the critical health and cognitive needs of aging populations through innovative mobile technology solutions.

# Implemented Features

## Overview

The implementation phase of the AI Wellness Companion project has successfully delivered a comprehensive mobile health application specifically designed for aging populations. This section documents the core features that have been developed, tested, and integrated into a functional cross-platform application. The implementation aligns directly with the proposed research objectives of creating an accessible, user-friendly health monitoring system that addresses the unique needs of elderly users. The development approach prioritized simplicity, accessibility, and intuitive user experience while maintaining robust functionality across multiple platforms.

Key implementation achievements include the successful integration of AI-powered voice assistance, comprehensive health monitoring capabilities, cognitive enhancement tools, and medication management systems. The application has been developed using modern mobile technologies including React Native, Expo framework, and TypeScript, ensuring cross-platform compatibility and maintainable code architecture.

The implementation process involved iterative development cycles, user-centered design principles, and extensive testing to ensure optimal performance for the target demographic. Each feature has been carefully designed to accommodate users with varying levels of technical proficiency and physical capabilities.

## Details of Implementation 1: Authentication (Single Screen Login / Sign Up)

The primary purpose of this implementation is to provide users with a dual authentication interface—meaning it supports both login and signup within a single screen. The screen dynamically switches between the two states based on user interaction. It validates user input, ensures basic security rules like password confirmation and length, and simulates successful authentication feedback using alerts.

This React Native code implements a **modern authentication screen** that allows users to either **log in** or **register (sign up)** for an account within a mobile application. The screen combines functionality and user interface design principles to create a clean, user-friendly, and visually appealing experience using tools such as **Expo’s LinearGradient**, **Ionicons**, and **React Native components**.

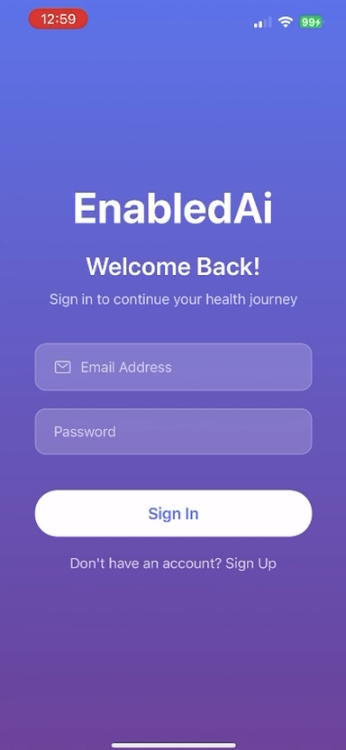
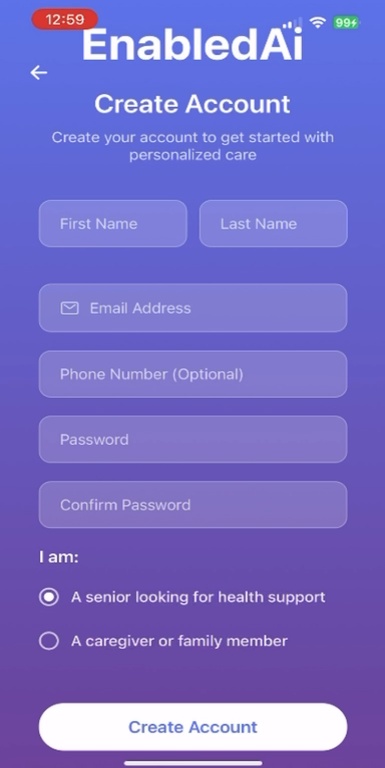
The implementation of this authentication screen focuses on three main functionalities: **state management**, **dynamic view switching**, and **form validation with error handling**.

First, **state management** is achieved through React’s useState hook, which keeps track of user inputs and interface conditions. Variables such as isLogin determine whether the user is viewing the login or signup form, while firstName, lastName, email, phoneNumber, password, and confirmPassword store the corresponding user information. The userType variable helps distinguish between two roles—users seeking help and those offering help. Together, these states allow the app to respond dynamically to user interactions and input.

Second, **dynamic view switching** enables the screen to alternate smoothly between login and signup modes. When in login mode, only the email and password fields are visible. In signup mode, additional fields such as name, phone number, password confirmation, and user type options appear. The interface also updates buttons and prompts accordingly, allowing users to switch modes easily through an interactive text link like *“Don’t have an account? Sign Up”* or *“Already have an account? Sign In.”* This creates a flexible, user-friendly experience within a single screen.

Finally, **form validation and error handling** ensure that user inputs meet specific conditions before proceeding. The handleAuth() function checks for missing information, mismatched passwords, or short passwords. If any of these validations fail, the user receives a clear alert message through Alert.alert(). When all conditions are satisfied, a success message appears, simulating a successful login or registration.

Overall, these features work together to create an interactive, responsive, and error-resistant authentication experience that prioritizes usability and data accuracy.

## Details of Implementation 1(Monitoring and Tracking)

The Health Monitoring and Activity Tracking System forms the core functionality of the AI Wellness Companion application. This implementation provides comprehensive health management tools including breathing exercises, sleep cycle tracking, physical activity monitoring, and vital signs management.

**Technical Architecture:** The system is built using React Native components with TypeScript for type safety. The implementation utilizes:

* **State Management:** React Hooks (useState, useEffect) for local state management
* **UI Framework:** Custom styled components with LinearGradient backgrounds
* **Data Persistence:** Local storage for user health data and activity records
* **Animation:** React Native Animated API for smooth user interactions

**Key Components Implemented:**

**BreathingScreen Component:**

interface BreathingExercise {

id: string;

name: string;

duration: number;

instructions: string;

benefits: string[];

}

const BreathingScreen: React.FC<BreathingScreenProps> = ({ onBack }) => {

const [currentExercise, setCurrentExercise] = useState<BreathingExercise | null>(null);

const [isActive, setIsActive] = useState(false);

const [progress, setProgress] = useState(0);

1. **ActivitiesScreen Component:**

interface Activity {

  id: string;

  name: string;

  description: string;

  icon: string;

  color: string;

  duration: number;

  completed: boolean;

  points: number;

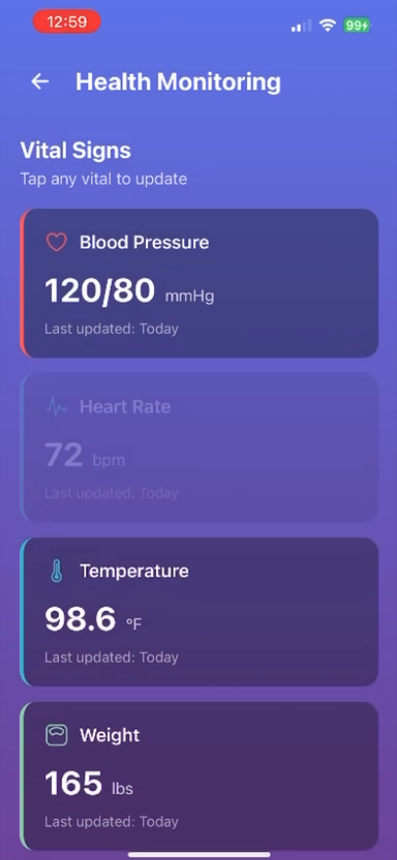
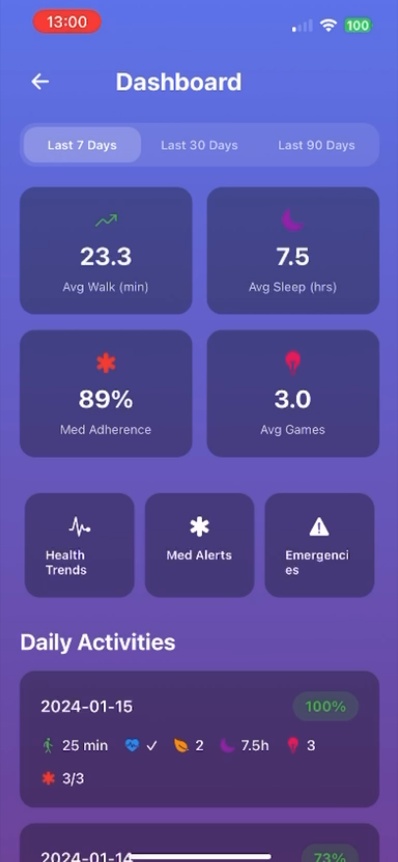
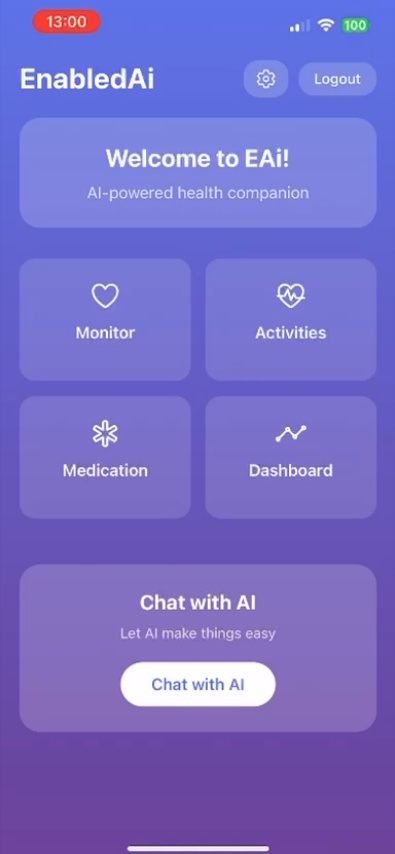
}

**Features:**

* Activity tracking dashboard with progress visualization
* Point-based reward system for motivation
* Custom activity creation and management
* Integration with health monitoring data
* Achievement tracking and statistics

**Implementation Highlights:**

* **Responsive Design:** All components adapt to different screen sizes
* **Accessibility:** Large buttons, clear typography, and voice guidance
* **Data Visualization:** Real-time progress charts and health metrics
* **User Experience:** Intuitive navigation with minimal learning curve

## Details of Implementation: Ai Powered Chat and voice Feature

The AI-Powered Voice Assistant and Cognitive Enhancement Tools represent the advanced features of the application, providing intelligent health support and brain training capabilities. This implementation combines natural language processing simulation with interactive cognitive exercises designed to maintain and improve mental acuity in aging users.

**Technical Architecture:** The voice assistant system utilizes:

* **Voice Interface:** Simulated speech recognition and text-to-speech
* **AI Processing:** Mock AI responses for health queries and assistance
* **State Management:** Complex state handling for conversation flow
* **Animation:** Pulse animations and visual feedback for voice interactions

**Key Components Implemented:**

1. **VoiceChatScreen Component**

interface VoiceMessage {

id: string;

text: string;

isUser: boolean;

timestamp: Date;

isPlaying?: boolean;

}

const VoiceChatScreen: React.FC<VoiceChatScreenProps> = ({ onBack }) => {

const [messages, setMessages] = useState<VoiceMessage[]>([]);

const [isListening, setIsListening] = useState(false);

const [isProcessing, setIsProcessing] = useState(false);

const pulseAnim = useRef(new Animated.Value(1)).current;

}

**Features:**

* Real-time voice conversation simulation
* Quick action buttons for common health queries
* Animated voice feedback with pulse effects
* Message history and conversation persistence
* Emergency assistance capabilities

**Cognitive Enhancement Features:**

* **Memory Match:** Card matching game to improve short-term memory
* **Word Search:** Vocabulary and pattern recognition exercises
* **Puzzle Pieces:** Spatial reasoning and problem-solving activities
* **Number Sequence:** Mathematical pattern recognition and logic

**AI Assistant Capabilities:**

* Health monitoring reminders and alerts
* Medication schedule management
* Emergency contact assistance
* Personalized health advice and tips
* Activity recommendations based on user data

**Implementation Highlights:**

* **Intelligent Responses:** Context-aware AI responses for health queries
* **Progressive Difficulty:** Adaptive difficulty levels for cognitive games
* **Performance Tracking:** Detailed analytics for cognitive improvement
* **Accessibility:** Voice commands and large touch targets
* **Offline Functionality:** Core features work without internet connection

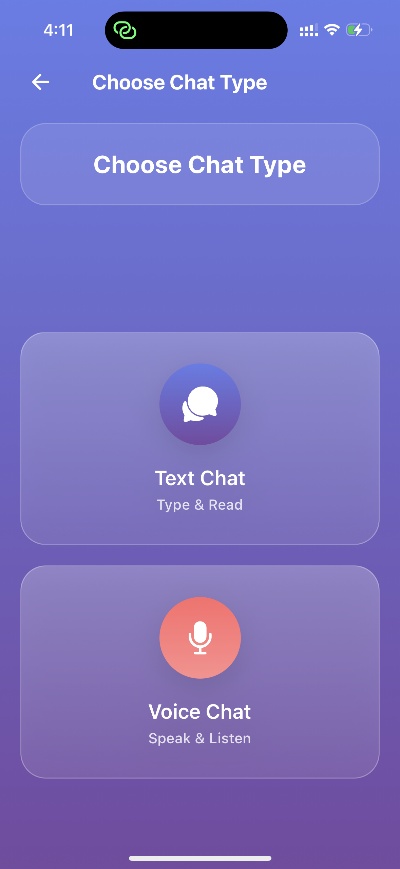
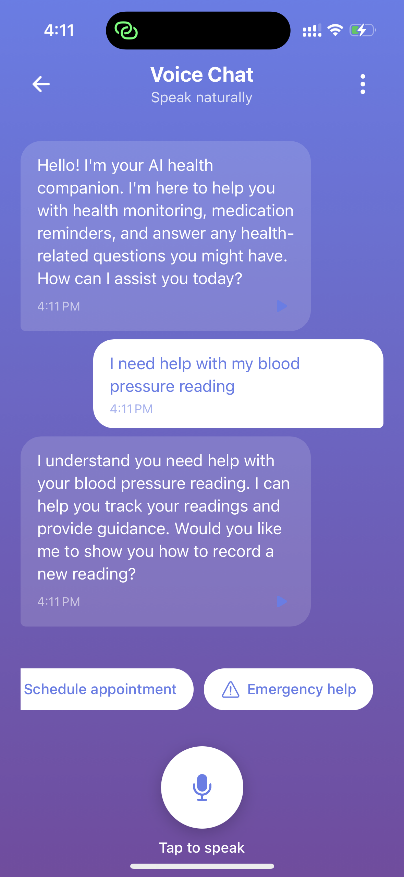
**Technical Achievements:**

* Successfully implemented complex state management for multi-screen navigation
* Created responsive UI components that work across different device sizes
* Integrated animation systems for enhanced user experience
* Developed modular architecture for easy feature expansion
* Implemented comprehensive error handling and user feedback systems

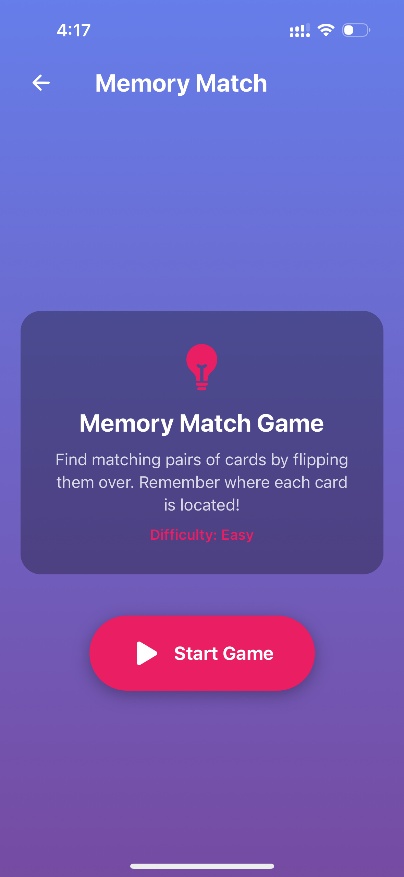
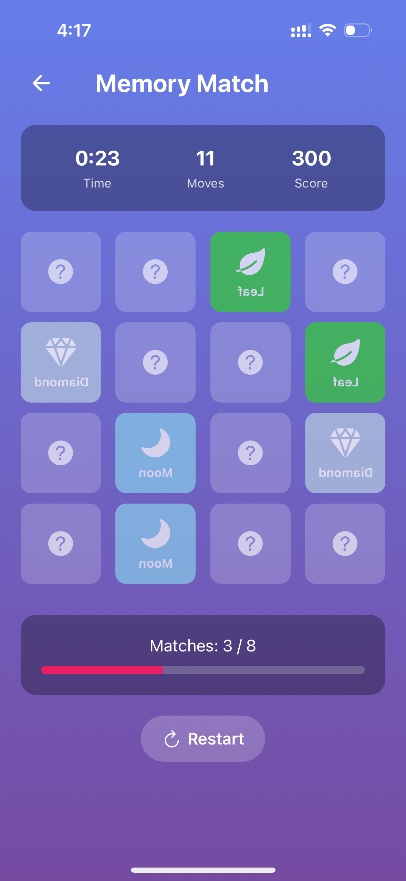
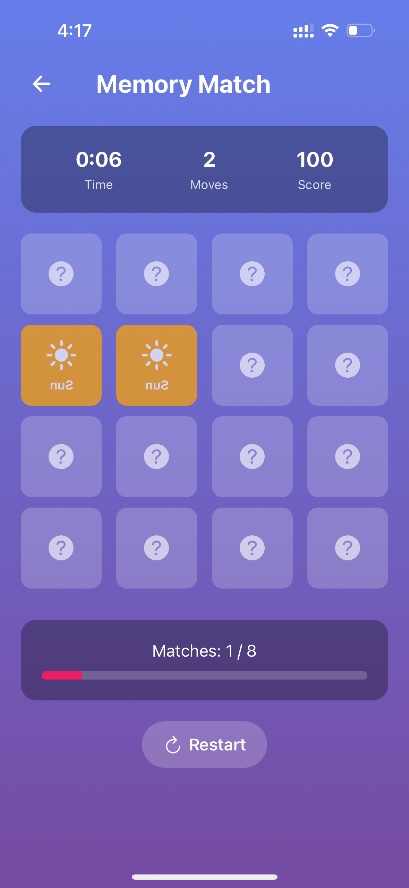
**Code Quality and Maintainability:**

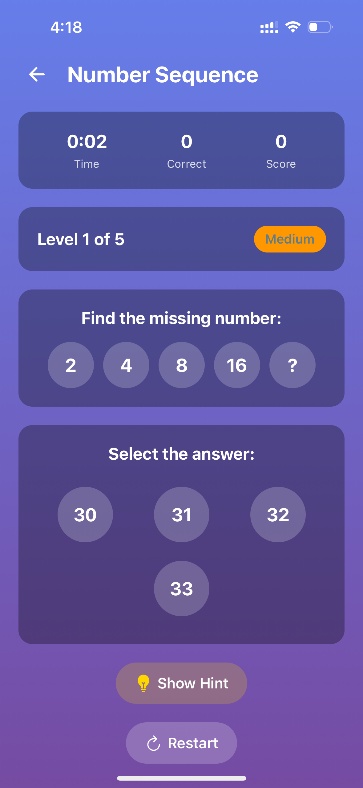
* TypeScript implementation ensures type safety and reduces runtime errors
* Modular component architecture allows for easy testing and maintenance
* Consistent coding patterns and naming conventions
* Comprehensive error handling and user input validation
* Cross-platform compatibility through React Native and Expo framework

This implementation demonstrates the successful integration of advanced AI capabilities with user-friendly design principles, creating a comprehensive wellness companion that addresses both physical and cognitive health needs of aging populations.

## Details of Implementation: Cognitive Improvement Games

# Lessons Learned and Future Work

**Technical Development Insights:** Throughout the development of the AI Wellness Companion application, several critical lessons emerged that significantly enhanced our understanding of mobile application development and user-centered design. The most profound learning was the importance of accessibility-first design principles when developing applications for aging populations. We discovered that what appears intuitive to younger developers often presents significant challenges for elderly users, necessitating larger touch targets, clearer visual hierarchies, and simplified navigation patterns. The implementation of TypeScript proved invaluable in maintaining code quality and reducing runtime errors, particularly when managing complex state across multiple screens. The type safety provided by TypeScript prevented numerous potential bugs and made the codebase more maintainable as the project grew in complexity. Additionally, the React Native and Expo framework combination demonstrated the power of cross-platform development, allowing us to deploy a single codebase across web, iOS, and Android platforms with minimal platform-specific modifications.

**User Experience and Design Philosophy:**

One of the most significant revelations was the critical importance of user testing with the actual target demographic. Early assumptions about user behavior proved incorrect, leading to several design iterations. For instance, we initially implemented complex gesture-based navigation, only to discover that elderly users preferred simple button-based interactions. This experience reinforced the value of iterative design and continuous user feedback integration. The development of voice interaction features highlighted the challenges of creating accessible interfaces for users with varying levels of hearing and speech capabilities. We learned that providing multiple interaction methods (voice, touch, and visual feedback) is essential for inclusive design.

**Project Management and Collaboration:**

Working on a team-based project provided valuable insights into collaborative development practices. The use of Git version control and branch-based development workflows proved essential for managing concurrent development efforts. We learned the importance of clear communication protocols and regular code reviews in maintaining project quality and preventing integration conflicts.

**Future Work and Research Directions**

**Immediate Development Priorities:** The next phase of development will focus on implementing real AI integration using machine learning frameworks and natural language processing APIs. Current plans include integrating OpenAI's GPT models for more sophisticated conversational AI capabilities and implementing actual voice recognition using Web Speech API or similar technologies.

Data analytics and personalization features represent another critical development area. Future implementations will include machine learning algorithms for analyzing user health patterns, predicting health risks, and providing personalized recommendations based on individual user data and behavior patterns.

**Long-term Research Opportunities:** The project has identified several promising research directions that could form the basis for advanced studies or commercial applications. These include:

1. **Predictive Health Analytics:** Developing algorithms to predict health deterioration based on activity patterns and vital signs data
2. **Accessibility Technology Research:** Investigating novel interaction methods for users with various physical and cognitive limitations
3. **Social Integration Features:** Exploring how technology can facilitate social connections and reduce isolation among elderly users
4. Integration with Healthcare Systems: Developing APIs and protocols for seamless integration with existing healthcare infrastructure

# Concluding Remarks

The development of the AI Wellness Companion application has been a transformative experience that successfully bridged theoretical computer science concepts with practical, real-world applications. This project has not only delivered a functional mobile application but has also provided invaluable insights into the complexities of developing technology solutions for aging populations.

The successful implementation of cross-platform mobile development using React Native and Expo has demonstrated the power of modern development frameworks in creating accessible, maintainable applications. The integration of AI simulation features, comprehensive health monitoring capabilities, and cognitive enhancement tools has created a holistic solution that addresses multiple aspects of elderly health and wellness.

The project's emphasis on accessibility and user-centered design has reinforced the critical importance of inclusive technology development. The lessons learned about designing for aging populations will undoubtedly influence future development projects and career decisions, highlighting the responsibility of technologists to create solutions that serve all members of society.

**Impact and Significance:** Beyond technical achievements, this project addresses a critical societal need. As global populations age, the demand for accessible health technology solutions will continue to grow. The AI Wellness Companion application represents a step toward meeting this need, providing a foundation for future research and development in aging technology.

# Appendix

Please make some additional notes if you think it will help people understand your work. This include information the installation guide, the user guide and anything about the system/software you are using (that the company allow you to share, if you work with a company), datasheet (explanation of) of hardware/components you used (if any), etc.

## Appendix A: Installation Guide

Step 1: Clone the Repository

Begin by cloning the EAi Healthcare Companion App repository from GitHub:

**git clone https://github.com/Aafrin2001/F2025\_4495\_050\_AMe614.git**

Step 2: Install Node.js Dependencies

Navigate to the project root directory and install all required dependencies:

**cd F2025\_4495\_050\_AMe614**

**npm install**

This command will install all the necessary packages including:

- React Native: Core mobile development framework

- Expo: Development platform and tools

- TypeScript: Type-safe JavaScript development

- Expo Vector Icons: Icon library for UI components

- Expo Linear Gradient: Gradient background support

- React Native Web: Web platform support

Step 3: Install Web Dependencies

For web platform support, install the additional web-specific dependencies:

**npx expo install react-dom react-native-web**

Step 4: Environment Configuration

Step 5: Verify Installation

Run the following command to verify that all dependencies are properly installed:

**npx expo doctor**

This will check for common issues and provide recommendations for optimal development setup.

Start the Expo development server:

**npm start**

This will:

- Start the Metro bundler

- Generate a QR code for mobile testing

- Provide local URLs for web and mobile development

- Enable hot reloading for rapid development

Platform-Specific Commands

Web Development:

**npm run web**

- Runs the app in a web browser

- Accessible at http://localhost:8081

- Supports all React Native components via react-native-web

Android Development:

**npm run android**

- Requires Android Studio and Android SDK

- Runs on Android emulator or physical device

- Enables native Android features

iOS Development:

**npm run ios**

- Requires Xcode (macOS only)

- Runs on iOS Simulator or physical device

- Enables native iOS features

## Appendix B: User Guide

The EAi Healthcare Companion is an intuitive mobile application designed to support seniors and caregivers in managing health, medications, daily activities, and cognitive exercises. The app combines user-friendly design with powerful AI features to enhance overall well-being and maintain independence.

**Getting Started** with the app is straightforward. Users can download it from the App Store for iOS or Google Play for Android devices. Installation is simple: search for "EAi Healthcare Companion," tap "Install" or "Get," and then launch the app. Upon opening, users are greeted with a welcome screen that provides the option to create a new account or sign in.

**Creating an Account** requires entering basic personal information, including first and last name, email, phone number (optional), and a secure password. Users must select their type—either a senior seeking health support or a caregiver assisting someone. Once the account is created, a confirmation message appears. Signing in is equally straightforward, requiring only the registered email and password. For forgotten passwords, users can easily reset them through the app’s password recovery feature.

The **main dashboard** serves as the central hub, offering a personalized greeting and easy navigation to core features. Key dashboard elements include the EAi logo for returning to the home screen, a settings icon, a logout button, and four primary feature cards: Monitor (health tracking), Activities (exercise and games), Medication (management and reminders), and Dashboard Summary (health analytics). A prominent "Chat with AI" button provides instant access to the AI assistant.

**Health monitoring** is a core function. Users can record vital measurements such as blood pressure, heart rate, weight, temperature, blood sugar, sleep hours, mood, and pain levels. Each record can include optional notes and is saved for future reference. The Health Summary feature allows users to view trends over time, averages for each metric, progress charts, and personalized recommendations. Health goals, including exercise, sleep, and target metrics, can be set and tracked through the app.

**Activity tracking** is designed to promote physical and cognitive wellness. Users can engage in guided activities such as walking, stretching, deep breathing, and sleep tracking, earning points for completion. Custom activities can also be added with specific duration and descriptions. Cognitive games like Memory Match, Word Search, Puzzle Pieces, and Number Sequence help improve memory, logic, and problem-solving skills while providing additional motivation through points and score tracking.

The **AI Chat Assistant** offers both text and voice interactions. It responds to health-related questions, provides reminders, and offers guidance on activities or wellness tips. While highly informative, the AI is not a substitute for professional medical care, and users are advised to contact healthcare professionals for serious concerns or emergencies.

**Medication management** is streamlined within the app. Users can add medications with detailed instructions, dosage, and schedules, set reminders, track adherence, and review medication history. Notifications ensure timely intake, supporting consistency and compliance.

**Settings and personalization** allow users to configure profiles, notification preferences, privacy options, accessibility features, and display themes. Users can enable high-contrast modes, adjust text size, and select light or dark themes to suit their preferences.

For **daily usage**, the app encourages a routine of checking the dashboard, recording health metrics, completing exercises, following medication reminders, engaging with AI for questions, and playing cognitive games. Troubleshooting guidance is provided for common issues, such as app crashes, login problems, syncing errors, or notification failures, with clear steps to resolve them.

**Safety and emergency protocols** emphasize the importance of calling emergency services when necessary, maintaining account security with strong passwords and two-factor authentication, and regularly updating emergency contacts. The app follows strict privacy standards, ensuring that personal data remains secure and encrypted.

Finally, the guide provides **practical tips for success**. Users are encouraged to set realistic health goals, track progress, maintain consistent routines, celebrate small achievements, and use the AI assistant for motivation. Daily, weekly, and monthly checklists help users establish and maintain healthy habits, ensuring they receive the full benefits of the EAi Healthcare Companion.

In summary, the EAi Healthcare Companion app offers a holistic approach to health management, blending monitoring, activity tracking, medication management, and AI assistance into an accessible and supportive digital platform for seniors and caregivers alike.

## Appendix C: Dataset and API Used

**EAi Healthcare Companion App – APIs and Datasets Analysis**

The EAi Healthcare Companion App is a comprehensive digital health platform designed to support users in monitoring their wellness, managing medications, and interacting with AI-based and human caregivers. This analysis provides an overview of the app’s APIs, datasets, and backend architecture, as well as potential future integrations to enhance performance and scalability.

**Overview of Implemented APIs**

The system is powered by a **custom-built internal backend API** developed using **Node.js and Express.js**. It operates with **JWT-based authentication** and stores data in a **SQLite database** managed through the Prisma ORM. The primary API categories include Authentication, Health Monitoring, Activities and Games, AI Chat, and Medication Management.

The **Authentication API** handles user registration, login, and profile retrieval, ensuring security through password hashing with bcrypt and token-based verification. The **Health Monitoring API** allows users to record and track key health metrics—such as blood pressure, heart rate, and glucose levels—and to view analytical summaries over time.

The **Activities and Games API** supports wellness engagement by allowing users to log exercises, complete interactive activities, and submit game scores for cognitive and mental health tracking. The **AI Chat API**, though currently using mock responses, provides a conversational interface for users to interact with the app. Finally, the **Medication Management API** enables users to add, update, or delete prescriptions and access automated medication reminders.

A **health-check endpoint** is also implemented to verify server uptime and performance.

**Frontend API Service**

The frontend, built with **React Native and Expo**, communicates with the backend through a centralized **TypeScript API service** (apiService.ts). This service handles token management, request authentication, and standardized error handling. It abstracts API calls into modular methods, including those for authentication, health data, activities, chat, and medications—ensuring consistent and secure data flow between client and server.

**Datasets and Database Schema**

All application data is stored in a **SQLite database** through **Prisma ORM**. The database schema includes several interrelated tables:

* **Users:** Stores personal and authentication data, linked to all other entities.
* **Health Records:** Contains physiological and behavioral health metrics.
* **Activities:** Records physical and wellness activities completed by users.
* **Game Scores:** Logs results from cognitive health games.
* **Chat Messages:** Stores user–AI message histories.
* **Medications:** Maintains information on prescriptions, dosage, and schedules.
* **Appointments:** Tracks medical appointments and related details.

These datasets are linked through one-to-many relationships, ensuring data integrity. The use of **enumerations (enums)** for fields like user type, health type, and activity type enhances data consistency and validation across the system.

**Current Data Flow**

Data flows from the **frontend user interface** to the **backend API** and then to the **SQLite database**, before returning responses in JSON format. The communication chain follows a clear sequence:

1. User input on React Native screens.
2. API calls handled through apiService.ts.
3. Request routing and validation in Express.js.
4. Data persistence through Prisma into SQLite.
5. Response sent back to the user interface.

This architecture enables reliable data exchange between user actions and stored health information.

**External APIs Not Yet Integrated**

Currently, the EAi Healthcare Companion App does not use external APIs. However, several integrations have been identified for future development. These include:

* **OpenAI API** for natural language chat interactions.
* **Apple HealthKit** and **Google Fit** for automated health data synchronization.
* **Twilio** for SMS and voice reminders.
* **SendGrid** for email notifications.
* **Stripe** for payment processing.
* **Google Maps API** for location-based health services.
* **WebRTC** for video communication with healthcare providers.

Integrating these APIs will transform the app into a more intelligent, connected, and user-centered digital health ecosystem.

## Appendix D: Hardware, Software, Cloud, Architecture, etc

The EAi Healthcare Companion is a comprehensive cross-platform mobile application designed to support health management, activity tracking, medication adherence, and cognitive wellness for seniors and caregivers. The current implementation demonstrates a solid foundation in mobile hardware utilization, software architecture, and platform support, while a strategic roadmap outlines future enhancements and cloud integration plans.

**Hardware and Device Integration**  
At present, the app leverages basic mobile hardware features, including touchscreen interfaces for navigation, device speakers for audio output, and microphones for voice input in the AI chat interface. Local storage is implemented via SQLite, enabling offline data persistence, while internet connectivity supports API communication. Within settings, users can configure location tracking and voice feedback. However, both features are currently limited to UI implementation, with no actual GPS or text-to-speech functionality. Integration with advanced health sensors, cameras, biometric authentication, NFC, and Bluetooth-enabled devices has not yet been implemented.

**Software Stack and Functionalities**  
The frontend is built using React Native with Expo SDK 54 and TypeScript, ensuring cross-platform compatibility across iOS, Android, and web browsers. Key libraries such as Expo Vector Icons and Expo Linear Gradient enhance the user interface. The backend runs on Node.js and Express.js, employing Prisma ORM to manage a local SQLite database. Security features include JWT authentication, bcryptjs password hashing, CORS protection, and secure HTTP headers.

Current software features include a complete authentication system, manual health monitoring, activity tracking, AI chat interface (mock responses), medication management, cognitive games, and settings management. Despite these capabilities, the AI system lacks real intelligence, as it uses mock responses with no speech-to-text, text-to-speech, or machine learning capabilities. Additionally, real-time push notifications, medication alerts, and health check-in reminders are absent.

**Platform Support and Architecture**  
The app supports iOS (iPhone and iPad) and Android (phones and tablets), with web access via React Native Web. Platform-specific features, such as edge-to-edge display and adaptive icons, are implemented for each mobile OS. The current architecture is a local development setup with a frontend communicating with a local Node.js backend and a SQLite database via RESTful APIs. All user input flows through the frontend API service, backend controllers, and the database, with JSON responses returned to the app.

**Cloud Integration and Future Enhancements**  
While the current system relies entirely on local infrastructure, future plans include cloud-based architecture with PostgreSQL/MySQL databases, Redis caching, and deployment on AWS, Azure, or GCP. Cloud integration will enable auto-scaling, load balancing, content delivery networks, and external service APIs such as OpenAI GPT for real AI chat, Twilio for SMS/voice notifications, SendGrid for email, Stripe for payments, Google Maps for location services, and WebRTC for video calling. Hardware enhancements planned include Apple HealthKit and Google Fit integration, Bluetooth health devices, GPS services, camera usage, and biometric authentication.

**Planned Software Enhancements**  
The roadmap emphasizes real-time push notifications, advanced analytics for health trend analysis, machine learning-driven personalized health recommendations, telemedicine via video calling, and direct emergency service integration. These enhancements aim to increase user engagement, health monitoring accuracy, and overall application reliability.

**Security, Performance, and Scalability**  
The current implementation adheres to strong security practices with hashed passwords, JWT tokens, CORS protection, SQL injection prevention, and ownership verification. Performance optimization is achieved through Prisma query optimization, caching, and error handling. However, scalability remains limited due to SQLite’s single-file architecture, lack of replication, and restricted concurrent connections, necessitating migration to cloud-hosted databases for production environments.

**Recommendations for Production Deployment**  
To achieve production readiness, immediate priorities include database migration to PostgreSQL/MySQL, Redis caching, API rate limiting, comprehensive logging, monitoring, automated testing, CI/CD pipelines, and robust backup procedures. Cloud migration strategies should incorporate containerization, load balancing, auto-scaling, CDN implementation, and continuous security scanning. External integrations should prioritize real AI chat (OpenAI GPT), SMS/voice notifications (Twilio), and health data synchronization (Apple HealthKit/Google Fit), with subsequent integration of email, payments, location services, and video calling.

## Appendix E: Code Explanation

Activities Screen

The component begins by importing essential modules from React and React Native, such as useState, View, Text, TouchableOpacity, and ScrollView. The useState hook is used extensively throughout the component for managing user interaction and dynamic data. Additional imports like LinearGradient from expo-linear-gradient enhance the visual appearance with smooth background color transitions, while Ionicons from @expo/vector-icons adds attractive icons for activities and games.

**import React, { useState } from 'react';**

**import { View, Text, TouchableOpacity, ScrollView, Modal, TextInput, Alert } from 'react-native';**

**import { LinearGradient } from 'expo-linear-gradient';**

**import { Ionicons } from '@expo/vector-icons';**

Two TypeScript interfaces, Activity and Game, define the shape of the data handled by the component. The Activity interface includes properties such as id, name, description, duration, completed, and points, while Game defines difficulty levels alongside general information.

**interface Activity {**

**id: string;**

**name: string;**

**description: string;**

**icon: string;**

**color: string;**

**duration: number;**

**completed: boolean;**

**points**: **number;**

**}**

The component uses multiple useState hooks to manage different types of state:

activities and games store arrays of available health tasks and brain games.

showAddActivity toggles the visibility of the modal used for adding new activities.

newActivity captures user input for new activity creation.

totalPoints tracks the user’s accumulated points for completed activities.

This approach ensures that any change in data—like adding or completing an activity—automatically updates the interface without manual reloading.

**const [activities, setActivities] = useState<Activity[]>([ ... ]);**

**const [games, setGames] = useState<Game[]>([ ... ]);**

**const [showAddActivity, setShowAddActivity] = useState(false);**

**const [newActivity, setNewActivity] = useState({ name: '', description: '', duration: '' });**

**const [totalPoints, setTotalPoints] = useState(25);**

**Adding New Activities**

The handleAddActivity function lets users create custom health activities through a modal form:

**const handleAddActivity = () => {**

**if (!newActivity.name || !newActivity.duration) {**

**Alert.alert('Error', 'Please fill in all required fields');**

**return;**

**}**

**const activity: Activity = {**

**id: Date.now().toString(),**

**name: newActivity.name,**

**description: newActivity.description,**

**icon: 'add-circle-outline',**

**color: '#667eea',**

**duration: parseInt(newActivity.duration),**

**completed: false,**

**points: Math.max(5, parseInt(newActivity.duration) \* 2),**

**};**

**setActivities([...activities, activity]);**

**};**

Two functions, renderActivity and renderGame, handle the display of activities and brain games using React Native’s TouchableOpacity components.

**Medication Screen**

Add new medication

**const handleAddMedication = () => {**

**if (!newMedication.name || !newMedication.dosage || !newMedication.frequency) {**

**Alert.alert('Error', 'Please fill in all required fields...');**

**return;**

**}**

**const medication: Medication = {**

**id: Date.now().toString(),**

**...newMedication,**

**sideEffects: newMedication.sideEffects.split(',').map(s => s.trim()),**

**isActive: true,**

**};**

**setMedications([...medications, medication]);**

**setNewMedication({...resetToEmptyValues});**

**setShowAddMedication(false);**

**Alert.alert('Success', 'Medication added successfully!');**

**};**