**ARIBO IFEOLUWA – 300389564**

**PROJECT PROPOSAL 2 SUMMARY (26th SEPTEMBER – 11th OCTOBER)**

Over the last two weeks, I have developed and implemented four major React Native components for the Health Companion application using TypeScript and Expo. The implementation includes a comprehensive Admin Dashboard for health monitoring, a Chat Selection interface, the main application screen with role-based navigation, and an interactive Memory Match game for cognitive exercise. All components feature modern UI/UX design with gradient backgrounds, responsive layouts, and state management using React hooks

| Date | Number of Hours | Description of Work Done |
|---|---|---|
| 26th September 2025 | 5.5 hours | - Scrum Meeting<br>- Implementation of Admin Dashboard Screen with TypeScript interfaces<br>- Setup of DailyActivity, HealthTrend, MedicationAlert, and EmergencyAlert data structures<br>- Research on state management patterns for complex dashboard components |
| 28th September 2025 | 2.5 hours | - -Continued Admin Dashboard development with summary grid implementation<br>- Added timeframe selector functionality (7d, 30d, 90d)<br>- Implemented modal system for health trends and alert displays<br>- Worked on responsive design using LinearGradient and percentage-based layouts |

| | | |
|---|---|---|
| 30th September 2025 | 1.5 hour | - Enhanced Admin Dashboard with real-time alert system<br>- Implemented severity-based color coding system<br>- Added medication stock monitoring with critical/low status indicators |
| 2nd October 2025 | 3.5 hours | - Developed Chat Selection Screen component<br>- Implemented dual-option card layout for text and voice chat<br>- Added gradient card backgrounds with icon containers<br>- Research on navigation prop patterns and callback implementations |
| 3rd October 2025 | 2 hours | - Created Main Screen component with role-based logic<br>- Implemented showAsEmployer and showAsProvider boolean flags<br>- Setup adaptive grid system for feature card rendering |
| 6th October 2025 | 1.5 hours | - - Enhanced Main Screen with dynamic layout switching<br>- Implemented conditional width assignment (100% vertical, 48% horizontal)<br>- Added feature cards for Health Monitoring, Activities, Medication, and Dashboard |
| 7th September 2025 | 2.5 hours | - Developed Memory Match Game Screen component<br>- Implemented card pairing system with 8 symbol types<br>- Added game state management using useState and useEffect hooks<br>- - Created flip animation simulation and match detection logic |
| 10th September 2025 | 2 hours | - Enhanced Memory Match game with scoring system<br>- Implemented real-time statistics display (time, moves, score) |

| | | |
|---|---|---|
| | | - Added progress bar and game completion alerts |
| 11th September 2025 | 4 hours | - Completed all four component integrations<br>- Performed comprehensive testing across all screens<br>- Finalized TypeScript interfaces and callback prop patterns<br>- Prepared technical documentation and repository organization |

## Description

On the frontend, I created four major screens using React Native with TypeScript. The Admin Dashboard Screen includes health monitoring with real-time alerts, medication tracking, and interactive data visualizations through modals. It supports timeframe analysis (7d/30d/90d), severity-based alerts, and detailed health trend tracking.

The Main Screen features a role-based layout that adjusts dynamically based on user roles (showAsEmployer/showAsProvider), using an adaptive grid system that switches between horizontal and vertical card arrangements. The Chat Selection Screen provides a clear choice between text and voice chat, with gradient backgrounds and smooth navigation flows.

I also built a complete Memory Match Game Screen, implementing full game mechanics including card pairing, move counting, time tracking, scoring, and progress display. The game uses eight distinct symbols and leverages React hooks for state management and interactive gameplay.

All components maintain consistent design patterns with LinearGradient backgrounds, responsive layouts, and TypeScript interfaces for type safety. Navigation uses callbacks, state management relies on useState and useEffect hooks, UI components are reusable and consistently styled, and data structures are rigorously modeled with TypeScript interfaces.

## Repository Check-in

I maintained a disciplined approach to version control throughout the development process, primarily working with the Main branch to ensure code stability and integration consistency. My repository management strategy included:

- **Primary Development Branch**: Exclusively used the Main branch for all stable development work
- **Pre-commit Protocol**: Always performed git pull origin main before pushing any changes to synchronize with the latest codebase and prevent merge conflicts
- **Incremental Commits**: Made frequent, focused commits with descriptive messages such as:
    - "Created and Added the Admin Dashboard Screen"
    - "Modified the MatchMemory screen" Etc..
- **File Organization**: Pushed all relevant source files including:
    - AdminDashboardScreen.tsx with complete health monitoring functionality
    - ChatSelectionScreen.tsx with dual-option interface
    - MainScreen.tsx with role-based navigation system
    - MemoryMatchScreen.tsx with complete game engine