



🌤️ Weather Dashboard - Project Instructions ## **COVER PAGE** --- #### **Application Name:**
Weather Dashboard #### **Group Number:** Group 14 #### **Platform:** React Native using Expo
CLI --- #### **Group Members** | **Role** | **Developer** | **Responsibilities** | |-----
|-----|-----| | **Developer 1** | **Anupa Ragoonanan** | API Integration &
Progress Management | | **Developer 2** | **Harry Joseph** | Interactive Components & Gestures | |
Developer 3 | **Raj Patel** | Modals, Notifications & Animations | | **Developer 4** | **Kerlan
Augustine** | Navigation & Project Documentation | --- ## **Phase 1** #### **A. Purpose of the
Application - Why is your application needed?** In an era of increasing climate variability and mobile-
first lifestyles, access to reliable, intuitive weather information has become essential for daily planning
and safety. While numerous weather applications exist, many suffer from poor user experience, limited
offline functionality, and lack of personalized interactivity. Our **Weather Dashboard** addresses these
critical gaps by providing: - **Enhanced User Experience:** Unlike static weather apps, our solution
features dynamic animations, gesture navigation, and responsive interactions that make weather
checking engaging rather than routine - **Reliability in Low Connectivity:** Many users experience
poor internet connectivity, especially students on campus or travelers. Our offline functionality ensures
weather data remains accessible when needed most - **Mobile-Optimized Design:** Built specifically
for React Native, delivering native performance and seamless cross-platform functionality -
Personalized Multi-Location Management: Essential for students, professionals, and families who
need to monitor weather across multiple locations for travel, commuting, or coordinating activities ####
B. What functionalities do you expect to implement? Our Weather Dashboard will deliver the
following core functionalities: **Primary Features:** - **Real-Time Weather Display:** Current
temperature, humidity, wind speed, and weather conditions using OpenWeatherMap API - **Extended
Forecast:** Detailed 7-day weather forecasts with daily highs, lows, and weather descriptions - **Multi-
Location Support:** Save, manage, and switch between unlimited weather locations with persistent
storage - **Offline Functionality:** AsyncStorage implementation to cache recent weather data for
access without internet connectivity **Interactive Features:** - **Touch Gestures:** Pull-to-refresh
functionality and swipe navigation between screens - **Dynamic Animations:** Smooth transitions and
weather-appropriate visual effects using React Native's Animated API - **Modal Interfaces:** Location
selection and detailed weather information overlays - **Push Notifications:** Weather alerts and daily
forecast updates **Technical Integration:** - **API Integration:** Secure OpenWeatherMap API calls
with error handling and data validation - **Cross-Screen Navigation:** React Navigation for seamless
user flow between home, details, and settings screens - **Local Storage:** User preferences and
location data persistence across app sessions #### **C. What is the general structure of your
application?** Our application follows React Native best practices with a modular, scalable architecture:
`` WeatherDashboard/ |—— App.js # Main application entry point |—— screens/ | |——
HomeScreen.js # Primary weather dashboard | |—— DetailsScreen.js # Extended forecast view | |——
SettingsScreen.js # Location and preference management |—— components/ | |—— WeatherCard.js #
Reusable weather display component | |—— LocationModal.js # Location selection interface | |——
NavigationBar.js # Bottom tab navigation | |—— LoadingSpinner.js # API loading indicators |——
services/ | |—— weatherAPI.js # OpenWeatherMap integration | |—— storage.js # AsyncStorage data
management |—— utils/ | |—— constants.js # App-wide configuration | |—— helpers.js # Utility
functions |—— styles/ |—— globalStyles.js # Consistent theming and styling `` **Architecture Flow:**
1. **App.js** initializes navigation and global state management 2. **HomeScreen** serves as the
primary interface displaying current weather and basic forecast 3. **DetailsScreen** provides
comprehensive 7-day forecasts and weather metrics 4. **SettingsScreen** manages saved locations and
user preferences 5. **Services layer** handles all external API communication and local data
persistence 6. **Components** provide reusable UI elements ensuring consistent user experience ####
D. How do you plan to distribute the functionalities among the group members? **Developer 1:**
Anupa Ragoonanan - API Integration & Data Management - OpenWeatherMap API integration with

comprehensive error handling - Weather data fetching, parsing, and state management - AsyncStorage implementation for offline capability - HomeScreen development and weather display components - Progress tracking and team coordination ****Developer 2: Harry Joseph - Interactive Components & User Experience**** - DetailsScreen implementation with extended forecast views - Interactive touch components and gesture recognition systems - Pull-to-refresh functionality and swipe navigation - Animation implementation using React Native's Animated API - Performance optimization for smooth user interactions ****Developer 3: Raj Patel - Modals, Notifications & Visual Polish**** - Modal component development for location selection and weather details - Push notification system setup and configuration - SettingsScreen creation and location management functionality - Micro-interactions and transition effects for enhanced user experience - Visual design consistency and UI/UX refinement ****Developer 4: Kerlan Augustine - Navigation & Project Integration**** - React Navigation setup and screen routing configuration - Application architecture design and component integration - Project management, timeline coordination, and quality assurance - Comprehensive documentation creation and maintenance - Testing coordination and final project presentation preparation ****Collaboration Strategy:**** - ****Weekly sync meetings**** to ensure consistent progress and address integration challenges - ****Shared component library**** to maintain design consistency across all screens - ****Git workflow management**** with feature branches and code reviews - ****Continuous integration testing**** to prevent breaking changes during development --- ****Weather Dashboard Development Team 14****
****CPAN 213 - Mobile App Development Project**** ****Phase 1 Submission - November 23, 2025****