MOBILE APPLICATION PROJECT / CONCEPT & IMPLEMENTATION

MyWeather Motion

A GPS-enabled, animated weather dashboard for Android



PROJECT PROPOSAL · VERSION 1.0 · 9 NOVEMBER 2026

Project field	Information
Project name	MyWeather / MyWeather Motion
Platform	Android, created with MIT App Inventor
Source project	MyWeather_PROJECT_RENAMED.aia
Current source version	4.0 (source version code 4)
Intended application ID	appinventor.ai_renzterrie.MyWeather
Project status	Prototype/project source available; final signed APK not yet supplied

Document note: This is a newly drafted proposal based on the available MyWeather source project, not a copy of a previously submitted or signed proposal. Confirm names, institution, adviser, grading criteria, and release dates before submission.

1. Executive summary

MyWeather Motion is a personal Android weather companion designed to make local and global weather conditions understandable at a glance. The application combines GPS-based location detection, a worldwide city search, an at-a-glance current conditions dashboard, and a three-day forecast. Motion graphics and a weather-inspired color palette provide a more engaging experience than a static information screen.

The initial implementation uses MIT App Inventor, its visual programming blocks, network components, local preference storage, and a bundled HTML/WebViewer interface. Forecast and geocoding information are requested from the Open-Meteo service.

2. Problem statement and rationale

Users frequently need quick, location-specific forecasts before commuting, studying, or planning outdoor activities. Weather applications can be crowded with data or make it difficult to change locations. MyWeather addresses these needs through clear weather indicators, convenient city lookup, and a simple mobile-first flow.

3. Goals and measurable objectives

- Present current temperature, weather condition, apparent temperature, humidity, wind, and precipitation after a successful lookup.
- Allow switching between GPS-based weather and manually selected cities.
- Display a three-day forecast, including precipitation probability and daily temperature range when API data is available.
- Support Celsius/Fahrenheit switching and preserve selected app preferences locally.
- Provide readable status, retry, and error states for missing permissions, invalid searches, and connection problems.
- Test essential workflows on Android devices and produce a release-ready build under the confirmed package name.

4. Target audience

Primary users include students, commuters, outdoor enthusiasts, travelers, and Android users who want straightforward weather updates. The interface is designed for quick interactions and should remain usable across common phone sizes.

5. Proposed scope

In scope	Outside initial scope
GPS-based current location and city search	Account registration and sign-in
Current conditions and three-day forecast	Severe-weather push alerts
Temperature-unit preference	Offline forecast synchronization
Weather animations and responsive UI	Subscription or payment features
Local saved preferences	Long-term weather history analytics

6. Functional requirements

ID	Requirement	Priority
FR-01	Request location permission and obtain coordinates when permitted.	High

FR-02	Search a named city and obtain its coordinates.	High
FR-03	Fetch and render the latest available forecast values.	High
FR-04	Show a three-day forecast and weather highlights.	High
FR-05	Refresh the currently selected location.	High
FR-06	Toggle temperature units and preserve the user choice.	Medium
FR-07	Show animation that reflects weather/diurnal status.	Medium
FR-08	Display actionable fallback and error messages.	High

7. Technical approach

Architecture: MIT App Inventor Screen1 handles controls and logic. LocationSensor provides device coordinates, Web components make Open-Meteo requests, TinyDB retains settings, and a local HTML file in WebViewer supplies animated visual elements. Responses are parsed into interface labels and forecast cards.

- Android development: MIT App Inventor visual blocks and UI components.
- Data source: Open-Meteo forecast API and geocoding lookup.
- Local storage: TinyDB for unit preference and selected city.
- Presentation: bundled HTML/CSS/JavaScript with WebViewer plus native App Inventor controls.
- Build/export: editable .aia project; signed Android APK for direct installation and signed AAB for Google Play distribution.

8. Non-functional requirements

- Usability: essential current conditions visible without complex navigation.
- Responsiveness: layouts scale cleanly across typical Android phone displays.
- Performance: animations should remain fluid and avoid blocking core controls.
- Reliability: gracefully handle API timeouts, unavailable internet, and denied GPS permission.
- Privacy: request location access only for weather lookup; document any stored preferences and app data.
- Security: distribute official signed builds and verify that package name and certificate are correct.
- Accessibility: use sufficient contrast, legible text, and meaningful status labels.

9. Implementation plan

Phase	Activities	Deliverable
1. Requirements	Validate users, data fields, UI and acceptance criteria.	Approved project plan
2. Development	Wire API requests, GPS, city search, preferences and UI.	Functional AIA project
3. Validation	Test installation, connectivity, permissions, units and forecasts.	Test results and defect log
4. Release	Confirm package, signing keys, Play assets and store policies.	Signed APK and AAB
5. Handover	Package source, documentation, and website downloads.	Project submission ZIP

10. Testing and acceptance criteria

- With location permission granted, GPS weather requests succeed and display a location-specific forecast when the network is available.
- With permission denied, the app remains usable through city search.
- A valid city search returns forecast information; a non-matching city produces a clear message.
- Refreshing initiates a new request and updates an observable status.
- Switching °C/°F updates relevant units and the preference remains after reopening.
- A signed APK installs on a supported test device and opens without critical errors.
- The final signed AAB has the exact Play Console package name and passes publishing checks.

11. Risks and mitigations

Risk	Mitigation
API/network unavailable	Display clear retry guidance, keep app responsive.
Location permission refused	Allow manual city search.
Incorrect Play package ID	Verify app ID before exporting a signed release.
Signing key mismatch	Preserve original keystore and verify certificate fingerprint.
App bundle rejected by Play	Check target SDK, package metadata, policies and release track requirements.

12. Deliverables and submission checklist

Item	Status / next action
Editable project	MyWeather_PROJECT_RENAMED.aia — available.
Raw source	Unpacked Screen1.scm, Screen1.bky, project.properties and embedded assets — available.
Whole-app archive	Source ZIP with the complete AIA contents — available.
Signed AAB	Only a *historical* signed AAB for the old app ID is available; it must not be represented as the new package release.
Signed release APK	Not available in the provided files; export from MIT App Inventor and confirm signing.
Documentation	This drafted proposal in DOCX/PDF plus the website PRD and TRD.

13. Review and approval

Prepared by: _____ Date: _____

Reviewed by: _____ Date: _____

Approved by: _____ Date: _____

Approval here is a submission placeholder; no approval is implied by this document.