Farmer Weather Notes (Offline First) – Week 2 Report

System Modeling & Design (ICT 381)

---

1. Introduction

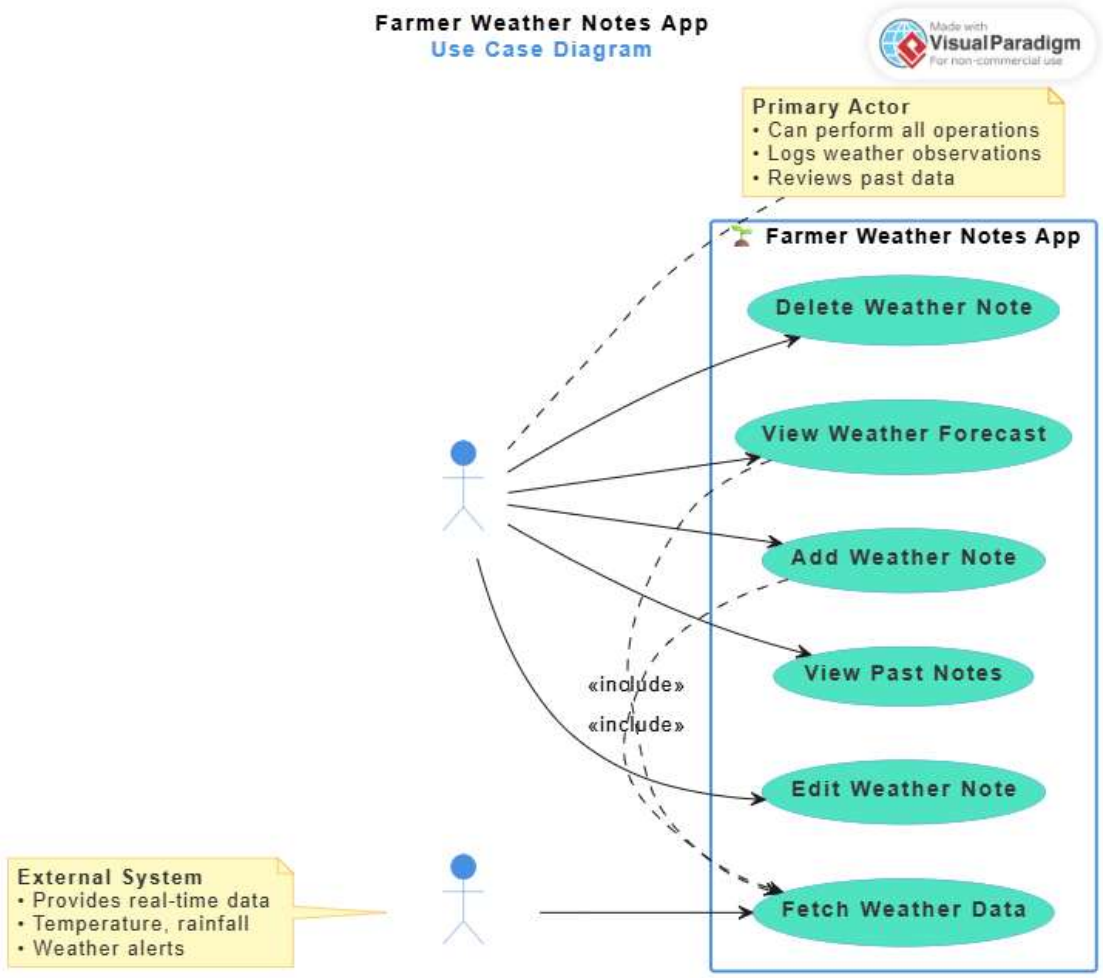
This phase of the project covers detailed system design through *use-case specifications* and *sequence diagrams*.
The goal is to describe interactions between the farmer (main user) and the system's internal components that handle data, logic, and storage, following an MVC (Model–View–Controller) structure.

---

2. Primary Actors and Use Cases

Primary Actor: Farmer
Supporting Actors: System (Controllers and Local Database), External Services (for sharing or backup).

Actor	Use Cases
Farmer	1. Add Farm Location 2. Record Daily Note 3. View Notes 4. Export/Share Notes 5. Receive Daily Reminder
System	6. Save and Sync Data (background operation)



3. Use Case Specifications

Use Case 1 - Add Farm Location

Primary Actor: Farmer
Preconditions: App open and storage available.
Trigger: User selects “Add Location.”
Main Flow:

System displays location form.

Farmer enters a name and optional coordinates.

System validates and saves to local storage

Confirmation message appears.

Alternate Flows: Duplicate name or GPS failure.
Postconditions: New LocationProfile stored offline.
Success Guarantee: Location available for note entry.

Use Case 2 - Record Daily Note

Primary Actor: Farmer
Preconditions: At least one LocationProfile exists.
Trigger: User taps “Add Note.”
Main Flow:

System opens Note form (date, weather, activity, comments).

Farmer enters data and saves.

Controller checks for existing note for same date/location.

If none, save to database; else prompt to edit or duplicate.

System confirms save/update.
Alternate Flows: Missing fields → prompt to complete; DB error → retry.
Postconditions: DailyNote record exists locally.
Success Guarantee: Exactly one note per location per date.

Use Case 3 - View Notes by Location/Season

Primary Actor: Farmer
Preconditions: Existing notes.
Trigger: User opens “View Notes.”
Main Flow:

Farmer selects filter (Location / Season / Date).

Controller queries local database.

System displays matching notes list.

Farmer opens details view.
Alternate Flows: No results → show “None found.”
Postconditions: Filtered notes displayed.
Success Guarantee: Accurate results shown offline.

Use Case 4 - Export or Share Notes

Primary Actor: Farmer
Preconditions: At least one note available.
Trigger: User taps “Export/Share.”
Main Flow:

Farmer chooses scope (single / filtered / all).

System formats data (.txt or .csv).

System launches device share sheet.

Farmer chooses method (e.g., email).

System confirms export.
Alternate Flows: No sharing apps → save locally; export failure → retry.
Postconditions: File created and shared/saved.
Success Guarantee: Data exported in selected format offline.

1.

Use Case 5 - Receive Daily Reminder

Primary Actor: Farmer
Preconditions: Reminders enabled; notification permission granted.
Trigger: Device time matches scheduled reminder.
Main Flow:

System shows notification.

Farmer taps notification.

App opens New Note screen.

Farmer records daily note.
Alternate Flows: Notifications off → in-app reminder; device off → missed alert shown next login.
Postconditions: Farmer prompted to record weather data.
Success Guarantee: Reminders trigger reliably each day.

1.

---

4. Sequence Diagram Summaries

Diagram 1 – Record Daily Note

Shows interaction between **Farmer → UI → Controller → Local DB**.
Steps: open form → enter data → save → validation → database write → confirmation.
Alt fragments cover duplicate note and missing field cases.

Diagram 2 – Export Notes

Participants: **Farmer**, **UI**, **ExportController**, **File Service**.
Sequence: user selects export → controller generates file → system returns share sheet → confirmation.

Diagram 3 – Receive Daily Reminder

Participants: **System Scheduler**, **Notification Service**, **Farmer**, **NotesActivity**.
Flow: schedule → trigger notification → tap → open New Note screen → save note.

**SEQUENCE DIAGRAM WAS PROVIDED IN A SEPARATE FILE

---

5. Conclusion

Week 2 refined the *Farmer Weather Notes* system design using structured use case specifications and UML sequence diagrams.
These artifacts clarify how the farmer interacts with controllers and storage layers, providing a foundation for Week 3's **activity** and **class diagram** development.