Farmer Weather Notes (Offline First) – Week 3 Report

1. Introduction

Week 3 focuses on modeling internal workflows, defining the class structure, and producing the 3-week Gantt schedule.
The objective is to refine system logic using **UML Activity Diagrams**, a **Class Diagram**, and a **Methodology Selection** that supports an MVC implementation strategy.

---

2. Activity Diagrams

Activity Diagram 1 – Record Daily Note Workflow

Start → Open App → Select Location → Enter Weather & Activity → Validate Input → [Decision]
• If data valid → Save to Local Database → Show Confirmation → End
• If invalid → Display Error → Re-enter Data → Validate again

Description:
This diagram shows the main user flow for logging a note. A decision node ensures input completeness before saving.

Activity Diagram 2 – Export Notes Process

Start → Open Notes List → Choose Export/Share → Select Scope → Generate File → [Decision]
• If export successful → Share/Save File → End
• If failed → Display Error → Retry or Cancel

Description:
Depicts the branching that occurs when an export succeeds or fails, emphasizing offline operation and retry logic.

---

3. Class Diagram Description

Key Classes:

Class	Attributes	Methods	Relationships
FarmerApp	version,	startApp(),	Aggregates

Class	Attributes	Methods	Relationships
	settings	showMainMenu()	Controllers
LocationProfile	locationId, name, coordinates	addLocation(), editLocation()	1..* to DailyNote
DailyNote	noteId, date, rainObs, activity, comments, locationId	saveNote(), editNote(), deleteNote()	Belongs to LocationProfile
NotesController	currentLocation, noteList	createNote(), validateNote(), getNotesByFilter()	Uses DailyNote and DatabaseHelper
ExportController	exportFormat	generateFile(), shareFile()	Depends on FileService
DatabaseHelper	dbPath	save(), update(), query(), delete()	Used by Controllers
ReminderService	reminderTime	scheduleReminder(), triggerNotification()	Communicates with FarmerApp

Relationships & Structure:

- **MVC pattern**: UI (Views) → Controllers → Models (LocationProfile, DailyNote) → DatabaseHelper.
- **Composition:** FarmerApp contains Controllers.
- **Association:** Each LocationProfile has many DailyNotes.
- **Dependency:** ExportController depends on FileService; ReminderService depends on OS scheduler.

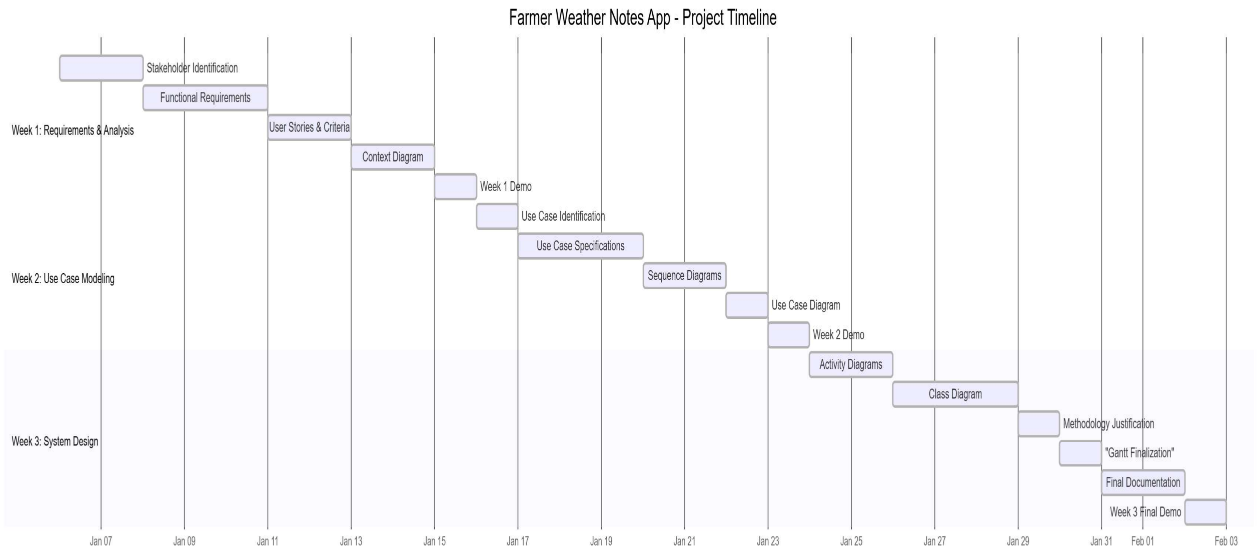
  -

---

4. Gantt Chart (3-Week Schedule)

Week	Activities	Deliverables	Duration
Week 1	Requirements Gathering & Analysis - Stakeholders, Vision, User Stories	Requirements Report, Context Diagram, PPT	1 Week
Week 2	Use Case & Sequence Diagrams - Identify Actors, Define Flows, Draw UML	Use Case Specs + Sequence Diagrams + PPT	1 Week
Week	Activity & Class Diagrams,	Activity Diagrams, Class	1 Week

Week	Activities	Deliverables	Duration
3	Gantt Chart, Methodology Justification	Diagram, Final Report, PPT	



5. Methodology Selection & Justification

Chosen Approach: *Iterative/Incremental Development*

Justification:

- The project involves ongoing refinement from analysis → design → implementation.
- Iterative cycles allow feedback after each week's deliverables, improving system accuracy and usability.
- The small student team benefits from short development sprints and early testing.
- Offline and local-storage components require frequent testing; an incremental method supports integrating those modules gradually.
- MVC structure fits incremental delivery: each iteration enhances one layer (Model, Controller, or View).

    -

6. Conclusion

Week 3 consolidates the *Farmer Weather Notes* design with detailed workflows, class structures, and a 3-week schedule.
The resulting models ensure consistent interaction between UI, controllers, and data layers—ready for prototype implementation in later phases.