Project Title: AgriBuddy – Smart Crop Scheduling & Advisory Web Application

NAME: Roshini Pininti

REGISTRATION NUMBER: 23BCE5129

Methodology Used: Scrum (Agile Framework)

1. Why Agile (Scrum) is Chosen

AgriBuddy consists of multiple interdependent modules such as user management, weather API integration, crop advisory logic, notifications, and analytics dashboard. These modules can be developed and tested incrementally. Therefore, Scrum is adopted as it supports iterative development, continuous integration, and milestone-based progress tracking.

Agile ensures incremental delivery, continuous validation of features, flexibility in refining advisory logic, and structured milestone monitoring. The total project duration is 10 weeks, divided into 4 sprints

2. Scrum Structure

2.1 Roles

Product Owner
The Product Owner defines system requirements, prioritizes backlog items, and validates sprint deliverables to ensure alignment with project objectives.

Scrum Master
The Scrum Master facilitates sprint planning, monitors sprint progress, ensures adherence to Agile principles, and resolves blockers during development.

Development Team
The development team is responsible for frontend development, backend API development, database design, API integration, advisory logic implementation, testing, and deployment. In an academic setting, these roles may overlap but are logically separated.

3. Product Backlog

The product backlog contains prioritized system features and technical tasks:

1. User Registration and Login
2. Farmer Profile Management
3. Crop Diary Module
4. Weather API Integration
5. Soil Data Handling
6. Crop Scheduling Logic
7. Advisory and Alert System
8. Notification System
9. Analytics Dashboard
10. Admin Panel
11. Testing and Deployment

Backlog refinement is performed before each sprint to ensure proper prioritization.

4. Sprint Planning

Sprint Duration: 2 weeks
Total Sprints: 4
Total Project Duration: 8 weeks

Each sprint includes sprint planning, development, testing, sprint review, and retrospective.

5. Sprint Breakdown and Milestones

Sprint 1 (Weeks 1–2)

Goal: Core Setup and User Management

Tasks:
Finalize requirements
Setup frontend and backend project structure
Design relational database schema
Implement user registration and login
Create farmer profile module

Deliverable:
Functional authentication system with profile management

Milestone 1:
User management module implemented and tested

Sprint 2 (Weeks 3–4)

Goal: Data Integration

Tasks:
Integrate Weather API
Create soil data input functionality
Design crop entity and database relationships
Connect frontend with backend APIs

Deliverable:
System successfully fetches and displays real-time weather data

Milestone 2:
Weather API integration and data flow validation completed

Sprint 3 (Weeks 5–7)

Goal: Advisory Logic and Notifications

Tasks:
Implement crop scheduling engine
Develop irrigation and fertilizer advisory rules
Implement crop diary logging
Add notification system (in-app/email)

Deliverable:
Automated advisory system based on weather and crop data

Milestone 3:
Advisory and alert system fully functional

Sprint 4 (Weeks 8-10)

Goal: Testing, Dashboard and Deployment

Tasks:
Develop analytics dashboard
Implement admin panel
Perform unit and integration testing
Fix identified bugs
Deploy application
Prepare documentation

Deliverable:
Complete working AgriBuddy web application

Milestone 4:
System deployed and ready for demonstration

6. Agile Ceremonies

Each sprint includes the following ceremonies:

Sprint Planning – Define sprint goals and tasks
Daily Stand-up – Monitor progress and address issues
Sprint Review – Demonstrate completed functionality
Sprint Retrospective – Identify improvements for next sprint

7. Incremental Product Versions

Version	Output
v0.1	User authentication module
v0.2	Weather integration module
v0.3	Advisory logic module
v1.0	Fully functional system

Each version incrementally enhances system capability.

8. Risk Management in Agile

API failure – Implement error handling and retry mechanisms
Development delays – Reprioritize backlog tasks
Integration errors – Perform testing after each sprint

9. Final Milestone Summary

Milestone	Description	Outcome
M1	User module completed	Login and profile system ready

Milestone	Description	Outcome
M2	API integration completed	Weather data functional
M3	Advisory system completed	Smart alerts operational
M4	Deployment completed	Full system available

Conclusion

The Agile Scrum model ensures structured, incremental development of AgriBuddy. Each sprint delivers a functional module, reducing risk, enabling early validation, and ensuring systematic progress toward full system deployment.

WORK BREAKDOWN STRUCTURE

Deliverable-Based Work Breakdown Structure

In AgriBuddy, each deliverable is a critical component of the complete system and represents a part of the working application. This structure helps in decomposing the full application scope into manageable submodules and ensures that each module aligns with the functional goals of the platform — such as crop recommendation, schedule generation, weather-based alerts, and user advisory.

The Level 1 deliverables of this WBS include:

- Requirement Specification Document
- Frontend Design (Web UI)
- Backend API Services
- Crop Schedule Logic Engine
- Weather & Advisory Module
- Database Design & Integration
- Notification System
- Testing & Validation
- Final Report and Submission

Each of these deliverables is further broken down:

- The Frontend Design deliverable includes wireframes, HTML/CSS layout, input forms for farm and crop details, and responsive design testing.
- The Backend API Services deliverable involves building Flask/Node.js APIs for CRUD operations, connecting farm/crop data to logic engines, and secure authentication handling.
- The Crop Schedule Logic Engine implements algorithmic recommendations based on crop type, region, and seasonal data.
- The Weather & Advisory Module connects to external APIs to fetch real-time weather data and generate dynamic advisory alerts.
- The Database Design & Integration task includes defining schemas for User, Farm,

Crop, ActivityLog, Notification, and connecting them to frontend and backend modules.

- The Notification System includes scheduled and event-triggered alert features via SMS/email or in-app.

This structure supports proper task ownership, ensures modular testing, and provides a clear breakdown of what the project will deliver. It also makes report writing and viva preparation easier as every component is documented as an individual output.

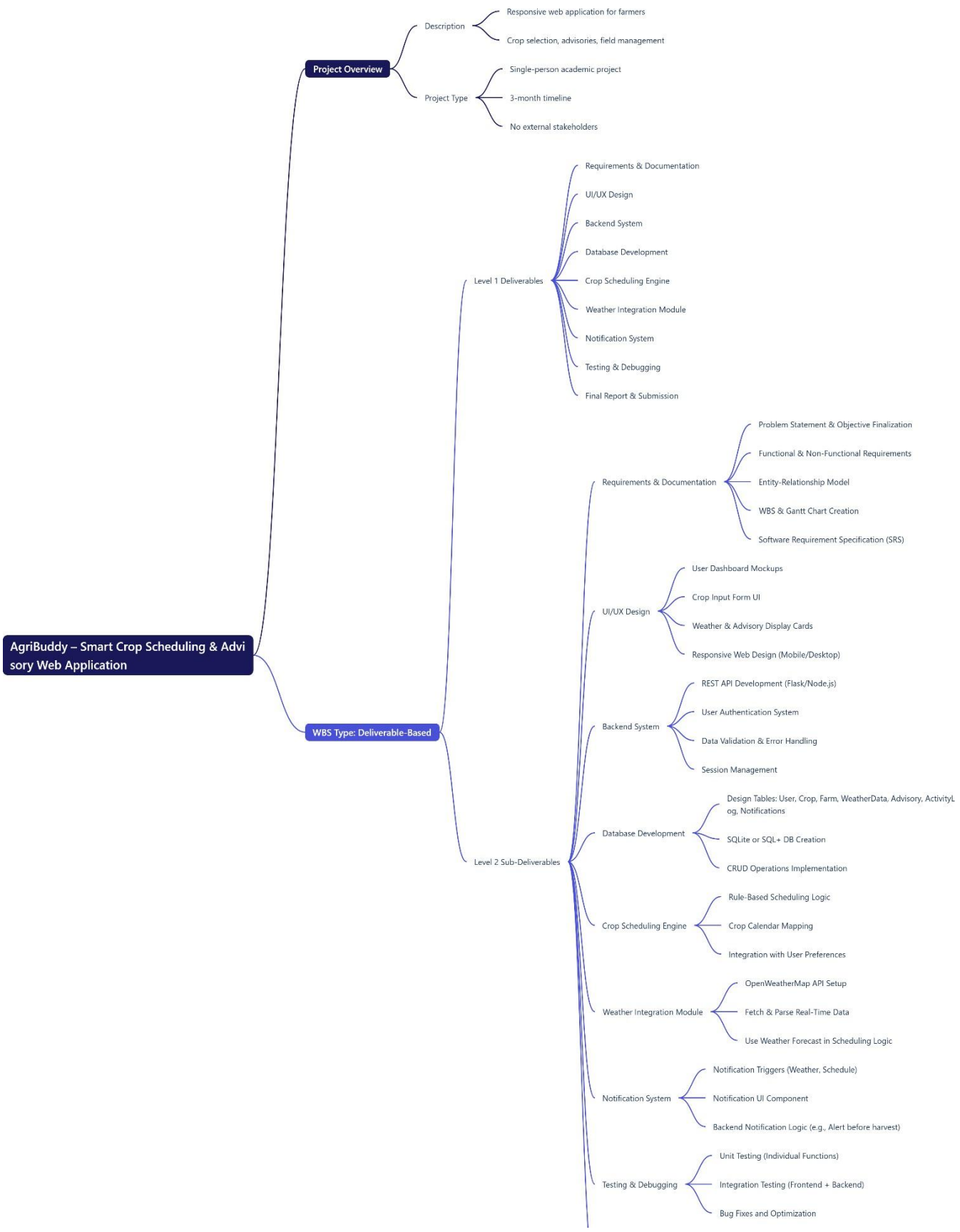

FIGURE 1 – DELIVERABLE BASED WORK BREAKDOWN STRUCTURE

Phase-Based Work Breakdown Structure

In a Phase-Based Work Breakdown Structure for the AgriBuddy project, the entire development cycle is broken into logical phases based on time and progression of work, typically followed in a structured academic project with a 3-month timeline. This structure emphasizes "when and how" the work is done, with deliverables grouped under each phase according to when they are to be completed.

For AgriBuddy, the Level 1 phases and their associated deliverables (Level 2) are:

🔹 **Phase 1: Planning & Requirement Analysis**

- Defining the project scope and objective
- Functional and non-functional requirement collection
- Preparing SRS document
- ER diagram and use-case modeling

🔹 **Phase 2: System & UI Design**

- Low-fidelity wireframes for user flows (farm entry, schedule view, etc.)
- Final UI/UX design of pages: Home, Login, Dashboard, Crop Advisory
- Frontend tech selection (HTML, CSS, JS, Bootstrap/Tailwind)
- Database schema design for entities: User, Farm, Crop, etc.

🔹 **Phase 3: Module Development**

- Implementing user login and authentication
- Building Farm, Crop, and ActivityLog CRUD features
- Integrating crop schedule logic (based on sowing/harvest rules)
- Developing advisory logic linked to weather APIs
- Backend and database connection setup (Flask + SQLite/PostgreSQL)

🔹 **Phase 4: Testing and Debugging**

- Frontend unit testing (form validation, alerts)
- Backend route/API testing using Postman
- Database integrity check (foreign key relations, data flows)
- Fixing UI/logic bugs
- Test run of crop schedule and alert notification triggers

🔹 **Phase 5: Deployment and Documentation**

- Deploying frontend and backend locally or on Heroku/Render

- Preparing final report: screenshots, outcomes, WBS & Gantt

This phase-wise structure makes it easier to monitor progress at each milestone, aligns with academic project reporting cycles, and simplifies risk management in case any module is delayed.

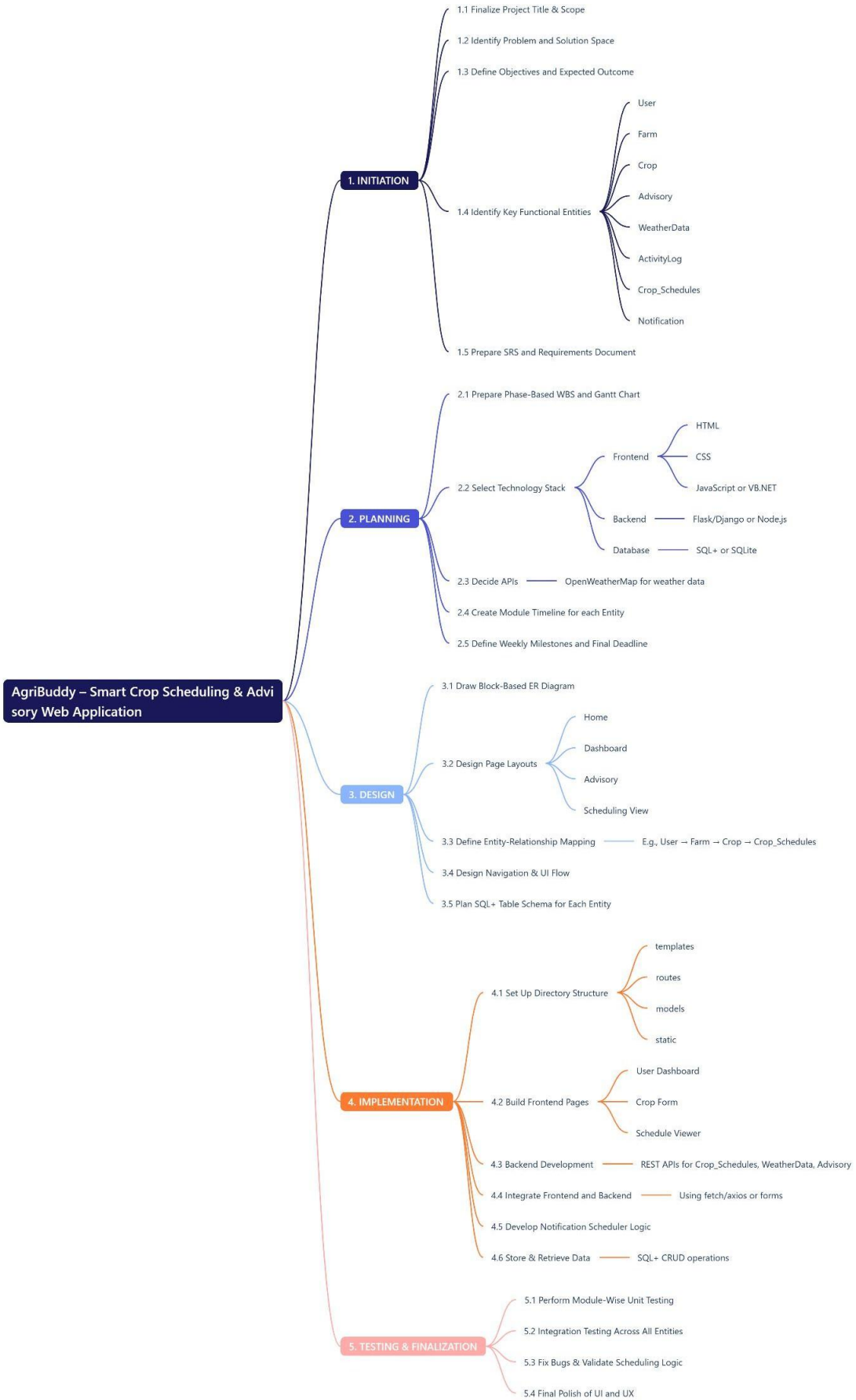

FIGURE 2 - PHASE BASED WORK BREAKDOWN STRUCTURE

Gantt Chart:

The Gantt Chart for the AgriBuddy project outlines the structured timeline for all major activities over a three-month development period, organized into sequential and overlapping phases. The chart begins with the Planning and Requirement Analysis Phase in Week 1, where project scope, objectives, and required functionalities such as crop scheduling, farm management, weather integration, and user advisory systems are identified and documented. This is followed by the UI & Database Design Phase, where low-fidelity wireframes are created for all screens including user dashboards, crop inputs, farm profiles, and weather data display, while in parallel, the database schema is finalized with entities such as User, Farm, Crop, WeatherData, Advisory, and their relationships. By Week 5, the Frontend Development Phase begins, focusing on implementing a responsive user interface using HTML, CSS, and JavaScript frameworks like React or Vue.js. Simultaneously, the Backend Development Phase is initiated in Week 6, where RESTful APIs are developed using technologies such as Node.js or Django to manage data operations, scheduling logic, and advisory generation. These phases run concurrently until Week 8 to maintain timeline efficiency. The Integration & Testing Phase spans Week 9 and Week 10, where unit testing, bug fixing, and frontend-backend integration are performed to ensure stable functionality. The Final Deployment & Documentation Phase occupies the last two weeks, where the application is hosted on a local server or cloud platform (e.g., Firebase or Heroku), and technical documentation, user manual, and final project reports are prepared. Each task in the Gantt chart is represented as a horizontal bar against a time axis, visually depicting dependencies, overlaps, and durations, offering a clear and efficient roadmap for single-developer academic execution.

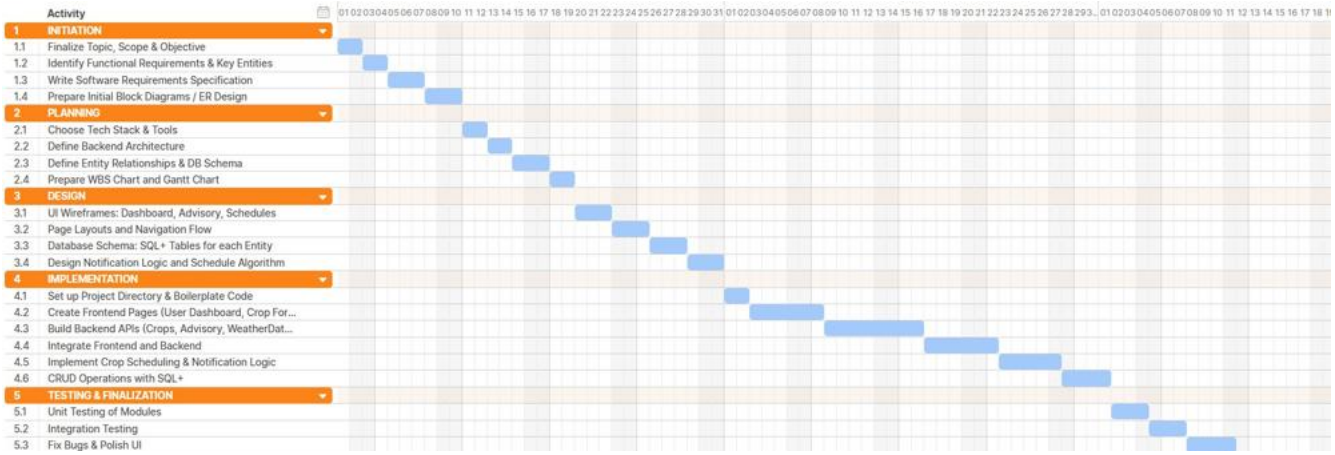

FIGURE-3 GANTT CHART