TEST PLAN

ASSIGNMENT-4

**1. Establish and Identify the Components**

**Components to Be Tested**

|  |  |
| --- | --- |
| Component | Description |
| User Registration & Login | Account creation, authentication, role-based access |
| AI Plant Analysis | Image upload, disease detection, and diagnosis accuracy |
| Fertilizer Recommendations | Accuracy of nutrient suggestions based on soil/plant data |
| Weather-Smart Planting | Forecast integration and correct crop recommendations |
| Security (Login + Data) | User data protection, secure access, and authorization control |
| AI Crop Assistant (Chatbot) | Responsiveness, accuracy, and reliability of AI answers |
| Agricultural Job Board | Posting, filtering, and matching logic |
| User Interface (UI/UX) | Navigation, usability, accessibility, and responsive design |
| Database & Backend API | Data storage, retrieval, and error handling |
|  |  |

**Component-wise Testing Approach**

* **Unit Testing (Per Component):**
  + Use mock data and testing frameworks (e.g., JUnit, PyTest, Postman) to test functions/methods individually.
  + Example: Test AI model outputs for known images, validate backend endpoints using mock requests.
* **Integration Testing (Multiple Components):**
  + Connect frontend to backend and validate real-time flows (e.g., registration -> login -> dashboard access).
  + Ensure modules like weather APIs or AI suggestions integrate smoothly with user-facing interfaces.
* **System Testing (Full Application):**
  + Perform end-to-end testing scenarios replicating actual user journeys.
  + Test system performance under load, multi-user scenarios, and across devices/browsers.

**2. Roles and Responsibilities**

| **Team Member** | **Testing Responsibility** | **Support Role** |
| --- | --- | --- |
| **Bikram Balal** | Unit testing of software modules (backend + frontend) | Integration testing support |
| **Sarmila Magar** | AI model testing (image analysis, prediction reliability) | Dataset preparation and performance evaluation |
| **Ramanjot Singh** | UI/UX testing (usability, accessibility) | Responsive design and cross-browser testing |
| **Aakarsh Kundukulam** | Validation of agricultural accuracy | Real-world test case inputs, recommendation reviews |
| **Bardan Poudel** | Test plan coordination, system and integration testing | Manages retesting and documentation |

**3. Document the Test Plan**

**List of Tests to Develop**

|  |  |  |  |
| --- | --- | --- | --- |
| Testing Type | What It Does | Why It’s Important | Tools Used |
| Unit Testing | Test individual components or functions (e.g., AI logic, login). | Ensures each part works correctly before combining. | JUnit, PyTest, Postman |
| Integration Testing | Test interactions between components (frontend, backend, APIs). | Ensures smooth communication and correct data flow. | Selenium, Postman, REST Assured |
| System Testing | Test the entire application as a whole. | Validates that the complete system meets requirements. | Manual Testing, Selenium |
| Usability Testing | Observe real users interacting with the platform. | Ensures the platform is user-friendly and intuitive. | Surveys, Observation, Maze |
| Security Testing | Test login, data protection, and vulnerabilities. | Prevents unauthorized access and data breaches. | OWASP ZAP, Postman, Burp Suite |
| Performance Testing | Check how the app performs under stress/load. | Ensures reliability during high usage. | JMeter, Locust |

**Test Development Timeframe**

| **Week** | **Task** |
| --- | --- |
| Week 1–2 | Unit tests for backend modules and AI components |
| Week 3 | Integration tests for UI/backend communication |
| Week 4 | System testing of full app and initial performance validation |
| Week 5 | UI/UX testing and accessibility reviews |
| Week 6 | Load testing and final bug fixes |

**Handling Failed or Inconclusive Tests**

* **Immediate Action:** Log bug in issue tracker (e.g., GitHub Issues).
* **Root Cause Analysis:** Assigned developer investigates cause.
* **Fix & Retest:** Component is updated and retested until success.
* **Version Control:** All fixes documented in Git for transparency.
* **Test Case Update:** Modify test case if failure revealed a flaw in logic.