🔄 Use Case Flows

Table of Contents

---

1. User Login

Actor: Farmer/User
Goal: Authenticate and access dashboard
Precondition: User has account credentials

Flow

1. User opens app → Login screen displayed
2. User enters email and password
3. User taps "Login" button
4. **App**: Shows loading indicator
5. **App**: Calls Firebase Auth login
6. **Firebase**: Validates credentials
7. **App**: Calls backend `/auth/login` with Firebase token
8. **Backend**: Validates token, returns app token + user info
9. **App**: Stores token in SharedPreferences
10. **App**: Navigates to Dashboard screen

PROF

Success

- User sees dashboard with "Welcome, [Name]"
- Token stored for future requests

Error Scenarios

- **Invalid credentials**: Show "Wrong email/password"
- **Network error**: Show "Check your connection"
- **Backend down**: Show "Service unavailable, try again"

---

2. View Dashboard

Actor: Logged-in User
Goal: See overview of farm/robot status
Precondition: User is authenticated

Flow

1. User sees Dashboard (home screen after login)
2. **App**: Shows loading skeleton
3. **App**: Calls `/landing` API
4. **Backend**: Returns:
    - Robot status (battery, location, active/idle)
    - Today's summary (weeds detected, area covered)
    - Recent alerts (low battery, detection spikes)
5. **App**: Displays data in cards/widgets
6. User scrolls to see all metrics

Success

- User sees:
    - Robot battery: 85%
    - Today: 127 weeds, 3.2 hectares
    - Alert: "Low herbicide level"

Error Scenarios

- **No data**: Show "No activity today"
- **API failure**: Show cached data + "Unable to refresh"

---

3. Check Robot Status

PROF

Actor: User
Goal: Get detailed robot health/location
Precondition: Robot is active or recently active

Flow

1. User taps "Robot Status" card on dashboard
2. **App**: Calls `/robot/status` API
3. **Backend**: Returns:
    - Battery: 85%
    - Location: GPS coordinates
    - Speed: 1.2 m/s
    - Status: Active/Idle/Charging
    - Last updated: timestamp
4. **App**: Shows map with robot pin + metrics

Success

- User sees robot location on map
- Metrics displayed clearly

Error Scenarios

- **Robot offline**: Show last known status + "Last seen 2 hours ago"
- **GPS unavailable**: Show "Location unknown"

---

4. View Weed Logs

Actor: User
Goal: See history of detected weeds
Precondition: Robot has detected weeds

Flow

1. User taps "Weed Logs" from bottom nav
2. **App**: Shows loading list
3. **App**: Calls `/weed-logs` API
4. **Backend**: Returns:
    - Summary: Total weeds per type (Broadleaf: 45, Grass: 82)
    - List of detections (timestamp, type, location, image)
5. **App**: Displays:
    - Pie chart of weed distribution
    - Scrollable list of detections
6. User taps a detection → sees full-screen image + details

Success

- User sees all weed detections chronologically
- Can filter by type or date

PROF

Error Scenarios

- **No detections**: Show "No weeds detected yet"
- **API failure**: Show cached logs + warning

---

5. Monitor Live Data

Actor: User
Goal: Watch real-time robot metrics
Precondition: Robot is active

Flow

1. User taps "Live Monitoring" from bottom nav
2. **App**: Shows dashboard with refreshing metrics
3. **App**: Calls `/monitoring/metrics` every 5 seconds
4. **Backend**: Returns:
    - Battery: 84% (decreasing)
    - Herbicide: 65%
    - Area covered: 3.25 hectares
    - Efficiency: 92%
5. **App**: Updates UI dynamically
6. **App**: Calls `/monitoring/activity` for timeline
7. **App**: Shows activity log (e.g., "10:32 AM - Started session")

Success

- User sees live-updating metrics
- Activity timeline shows recent events

Error Scenarios

- **Robot idle**: Show "Robot not active"
- **Connection lost**: Show "Reconnecting..." + last known data

---

6. Check Weather & Soil

Actor: User
Goal: View environmental conditions
Precondition: Sensors are active

Flow

1. User taps "Weather" from bottom nav
2. **App**: Calls `/environment` API
3. **Backend**: Returns:
    - Current weather (temp, humidity, conditions)
    - 7-day forecast
    - Soil data (temp, pH, moisture)
    - Today's recommendations ("Good day for spraying")
4. **App**: Displays:
    - Weather cards with icons
    - Forecast list
    - Soil metrics with gauges
    - Recommendation banner

PROF

Success

- User sees current: 28°C, Sunny, Humidity 45%
- Forecast shows next 7 days

- Soil: pH 6.5, Temp 22°C

Error Scenarios

- **Sensor offline**: Show "Soil data unavailable"
- **Weather API down**: Show last cached forecast

---

7. Generate Reports

Actor: User
Goal: View farming performance metrics
Precondition: Historical data exists

Flow

1. User taps "Reports" from bottom nav
2. **App**: Calls `/reports` API
3. **Backend**: Returns:
    - Widgets (total weeds, area, herbicide used, efficiency)
    - Trend chart (weeds detected over 7 days)
    - Distribution chart (weed types per crop)
4. **App**: Displays interactive charts
5. User taps "Export" → selects PDF or CSV
6. **App**: Calls `/reports/export?format=pdf`
7. **Backend**: Generates report, returns download link
8. **App**: Downloads file to device

Success

- User sees visual reports
- Can export as PDF/CSV

PROF

Error Scenarios

- **No data**: Show "Not enough data for report"
- **Export fails**: Show "Download failed, try again"

---

8. Browse Image Gallery

Actor: User
Goal: View weed detection images
Precondition: Images have been captured

Flow

1. User taps "Gallery" from bottom nav
2. **App**: Calls `/gallery` API

3. **Backend**: Returns list of image thumbnails with metadata
4. **App**: Displays grid of images
5. User scrolls, lazy-loads more images
6. User taps image → full-screen view
7. User can swipe between images
8. User taps "Delete" → confirms
9. **App**: Calls `DELETE /gallery/:id`
10. **Backend**: Deletes image
11. **App**: Removes from grid

Success

- User browses all weed photos
- Can view full-screen and delete

Error Scenarios

- **No images**: Show "No images yet"
- **Load failure**: Show "Unable to load images"
- **Delete fails**: Show "Deletion failed"

---

9. Ask Assistant

Actor: User
Goal: Get farming advice via chatbot
Precondition: User has questions

Flow

1. User taps "Assistant" from bottom nav
2. **App**: Shows chat interface
3. **App**: Calls `/assistant/history` to load past messages
4. User types question: "Why are weeds increasing?"
5. User taps send
6. **App**: Shows typing indicator
7. **App**: Calls `POST /assistant/query` with message
8. **Backend**: Processes query (AI/rule-based)
9. **Backend**: Returns answer
10. **App**: Displays assistant response in chat bubble
11. Chat history updated

Success

- User gets relevant farming advice
- Chat history persists

Error Scenarios

PROF

- **API timeout**: Show "Assistant not responding"
- **Invalid query**: Show "I don't understand, try rephrasing"

---

10. Manage Profile

Actor: User
Goal: Update personal/farm information
Precondition: User is logged in

Flow

1. User taps "Profile" from bottom nav
2. **App**: Calls `/profile` API
3. **Backend**: Returns user info, farm info, settings
4. **App**: Displays profile screen with editable fields
5. User taps "Edit Name" → enters new name
6. User taps "Save"
7. **App**: Calls `PUT /profile` with updated data
8. **Backend**: Updates database
9. **Backend**: Returns success
10. **App**: Shows "Profile updated" toast
11. User taps "Farm Info" → edits farm size
12. **App**: Calls `PUT /profile/farm` with new data
13. **Backend**: Updates farm info
14. **App**: Refreshes profile screen

Success

- User sees updated profile immediately
- Changes persist after logout/login

PROF

Error Scenarios

- **Validation error**: Show "Name cannot be empty"
- **Network error**: Show "Update failed, try again"
- **Unauthorized**: Re-login required

Common Patterns Across Use Cases

Loading States

- Show skeleton loaders or progress indicators
- Disable user actions during loading

Error Handling

- Display user-friendly error messages

- Offer retry options
- Cache data when possible for offline viewing

Authentication

- All API calls include auth token via AuthInterceptor
- Token refresh handled automatically
- Session timeout redirects to login

Real-time Updates

- Use polling (every 5-10s) for live data
- FCM notifications for urgent alerts
- Pull-to-refresh on all list screens

Data Persistence

- Store auth token in SharedPreferences
- Cache API responses for offline access (future)
- Clear cache on logout