Inventory App Agile Plan

Introduction

This Agile plan outlines the development roadmap for the Inventory App, a cross-platform application for managing inventories across domains like warehouse, retail, and personal collections. The app includes web (Vue.js), mobile (React Native, with native Swift for iOS and Kotlin for Android), and backend (Express.js, MongoDB) components. The plan incorporates core CRUD functionality and 12 enhancements (authentication, search, offline support, WebSockets, barcode scanning, analytics, internationalization, CI/CD, performance, IoT, accessibility, and native mobile transition) as specified in the comprehensive design document. Using the Scrum framework, development is divided into 12 two-week sprints (14 project days each, totaling 168 project days), plus 13 days for final testing and deployment, for a total of 181 project days.

Project Scope

- **Core Features**:
  - CRUD operations (add, view, edit, delete items).
  - Web frontend with Vue.js, modals, and TailwindCSS.
  - Mobile frontend with React Native, transitioning to Swift (iOS) and Kotlin (Android).
  - Express.js REST API with MongoDB.
- **Enhancements**:
  - Authentication, search/filtering, offline support, WebSockets, barcode scanning, analytics dashboard, internationalization, CI/CD, performance optimization, IoT integration, accessibility, native mobile transition.
- **Timeline**: 181 project days (168 for sprints, 13 for final testing/deployment).
- **Team**: 10 members (2 frontend, 2 backend, 2 Swift iOS, 2 Android, 1 QA, 1 DevOps).

Agile Methodology

- **Framework**: Scrum.
- **Sprint Duration**: 14 project days (2 weeks).
- **Ceremonies**:
  - **Sprint Planning**: Define sprint goals and backlog (2 hours).
  - **Daily Standups**: 15-minute sync on progress/blockers.
  - **Sprint Review**: Demo features, gather feedback (2 hours).
  - **Sprint Retrospective**: Reflect on process improvements (1 hour).
- **Artifacts**:
  - **Product Backlog**: All features/enhancements.
  - **Sprint Backlog**: Selected items per sprint.

  - **Increment**: Potentially shippable product at sprint end.

Team Structure

- **Frontend Developers (2)**: Specialize in Vue.js, TailwindCSS, and web accessibility.
- **Backend Developers (2)**: Focus on Express.js, MongoDB, and API integrations (WebSockets, IoT).
- **Swift iOS Developers (2)**: Develop native iOS app with SwiftUI, handle iOS-specific features (e.g., Core Data).
- **Android Developers (2)**: Develop native Android app with Jetpack Compose, handle Android-specific features (e.g., CameraX).
- **QA Engineer (1)**: Conducts unit, integration, and end-to-end testing.
- **DevOps Engineer (1)**: Manages CI/CD pipeline and deployments.

Epics

The project is divided into six epics:

1. **Core Inventory Functionality**: CRUD operations, basic UI, API.
2. **Security and Authentication**: JWT-based authentication, RBAC.
3. **Advanced Features**: Search, offline support, WebSockets, barcode scanning, analytics, internationalization.
4. **Performance and Scalability**: Pagination, caching, optimization.
5. **Integrations and Accessibility**: IoT integration, accessibility enhancements.
6. **Native Mobile Transition and CI/CD**: Swift iOS, Kotlin Android, automated testing/deployment.

Product Backlog

The backlog lists user stories, acceptance criteria, story points (Fibonacci: 1, 2, 3, 5, 8), and MoSCoW prioritization (Must-have, Should-have, Could-have, Won't-have).

Epic 1: Core Inventory Functionality

- **User Story 1.1**: As a user, I want to add an item to the inventory so I can track new stock.
  - **Acceptance Criteria**:
    - Form with name (required), quantity (required, 0), location (optional), description (optional).
    - POST /api/items saves item to MongoDB.
    - Success message displayed.
  - **Story Points**: 5
  - **Priority**: Must-have

- **User Story 1.2**: As a user, I want to view a list of items so I can see current inventory.
  - **Acceptance Criteria**:
    - GET /api/items returns item list.
    - Web: Rendered in ItemList.vue with ItemCard.vue.
    - Mobile: Displayed in FlatList (React Native).
  - **Story Points**: 3
  - **Priority**: Must-have
- **User Story 1.3**: As a user, I want to edit an item so I can update details.
  - **Acceptance Criteria**:
    - GET /api/items/:id fetches item.
    - PUT /api/items/:id updates item.
    - Pre-filled form in EditItemModal.vue and EditItem.js.
  - **Story Points**: 5
  - **Priority**: Must-have
- **User Story 1.4**: As a user, I want to delete an item so I can remove obsolete stock.
  - **Acceptance Criteria**:
    - DELETE /api/items/:id removes item.
    - Confirmation prompt before deletion.
  - **Story Points**: 3
  - **Priority**: Must-have
- **User Story 1.5**: As a developer, I want a MongoDB schema for items so data is structured.
  - **Acceptance Criteria**:
    - Schema includes name, quantity, location, description, sku, category, createdAt, updatedAt.
    - Text indexes on name, location; sparse index on sku.
  - **Story Points**: 2
  - **Priority**: Must-have

Epic 2: Security and Authentication

- **User Story 2.1**: As a user, I want to log in so only authorized users access the app.
  - **Acceptance Criteria**:
    - POST /api/auth/login returns JWT.
    - Login forms in Vue.js (Login.vue), React Native (Login.js).
    - Token stored securely (localStorage, AsyncStorage).
  - **Story Points**: 8
  - **Priority**: Must-have
- **User Story 2.2**: As an admin, I want role-based access so I can restrict actions.
  - **Acceptance Criteria**:
    - Roles: admin, manager, viewer.
    - Middleware restricts DELETE to admins.

    - UI hides restricted actions.
  - **Story Points**: 5
  - **Priority**: Must-have

Epic 3: Advanced Features

- **User Story 3.1**: As a user, I want to search and filter items so I can find stock quickly.
  - **Acceptance Criteria**:
    - GET /api/items?name=X&category=Y supports filtering.
    - Search/filter UI in HomePage.vue and HomePage.js.
  - **Story Points**: 5
  - **Priority**: Should-have
- **User Story 3.2**: As a mobile user, I want to work offline so I can manage inventory without internet.
  - **Acceptance Criteria**:
    - Local storage (IndexedDB, AsyncStorage).
    - POST /api/sync handles batched updates.
    - Background sync when online.
  - **Story Points**: 8
  - **Priority**: Should-have
- **User Story 3.3**: As a user, I want real-time updates so I see team changes instantly.
  - **Acceptance Criteria**:
    - Socket.io broadcasts create/update/delete events.
    - Clients update UI in real-time.
  - **Story Points**: 5
  - **Priority**: Should-have
- **User Story 3.4**: As a mobile user, I want to scan barcodes so I can add/update items quickly.
  - **Acceptance Criteria**:
    - GET /api/items/by-sku retrieves item.
    - Scanning via getUserMedia (web), vision-camera (React Native).
  - **Story Points**: 5
  - **Priority**: Should-have
- **User Story 3.5**: As a manager, I want an analytics dashboard so I can monitor stock trends.
  - **Acceptance Criteria**:
    - GET /api/analytics/summary returns category totals.
    - Charts in Dashboard.vue and Dashboard.js.
  - **Story Points**: 5
  - **Priority**: Could-have
- **User Story 3.6**: As a user, I want multi-language support so I can use my native language.
  - **Acceptance Criteria**:
    - UI translations (vue-i18n, i18next).

    - Item translations in schema.
  - **Story Points**: 5
  - **Priority**: Could-have

Epic 4: Performance and Scalability

- **User Story 4.1**: As a user, I want fast item loading so I can work efficiently.
  - **Acceptance Criteria**:
    - Pagination in GET /api/items.
    - Redis caching for queries.
    - Virtual scrolling in Vue.js, optimized FlatList in React Native.
  - **Story Points**: 5
  - **Priority**: Should-have

Epic 5: Integrations and Accessibility

- **User Story 5.1**: As a warehouse manager, I want IoT integration so I can track items via RFID.
  - **Acceptance Criteria**:
    - POST /api/iot/rfid updates quantities.
    - Mobile apps connect to RFID scanners.
  - **Story Points**: 8
  - **Priority**: Could-have
- **User Story 5.2**: As a user with disabilities, I want accessible interfaces so I can use the app.
  - **Acceptance Criteria**:
    - WCAG-compliant UI (ARIA, accessibilityLabel).
    - Tested with VoiceOver, TalkBack.
  - **Story Points**: 5
  - **Priority**: Should-have

Epic 6: Native Mobile Transition and CI/CD

- **User Story 6.1**: As a mobile user, I want a native iOS app for a fast, authentic experience.
  - **Acceptance Criteria**:
    - SwiftUI app with core CRUD.
    - Integrates with API via URLSession.
  - **Story Points**: 8
  - **Priority**: Could-have
- **User Story 6.2**: As a mobile user, I want a native Android app for a fast, authentic experience.
  - **Acceptance Criteria**:
    - Jetpack Compose app with core CRUD.
    - Integrates with API via Retrofit.
  - **Story Points**: 8

  - **Priority**: Could-have
- **User Story 6.3**: As a developer, I want a CI/CD pipeline to automate testing/deployment.
  - **Acceptance Criteria**:
    * GitHub Actions for Jest, Vitest, Cypress, Detox.
    * Deploys web to Netlify, backend to AWS.
  - **Story Points**: 5
  - **Priority**: Should-have

Sprint Plan with Detailed Task Breakdown

Each sprint lasts 14 project days, with tasks broken down into sub-tasks, estimated hours, and assigned roles. Total hours are estimated assuming a 40-hour workweek per developer (80 hours per pair), with QA and DevOps contributing as needed.

Sprint 1 (Day 1–14): Core Backend and Database

- **Goal**: Set up Express.js API and MongoDB schema.
- **Stories**: 1.5 (MongoDB schema, 2 points), 1.1 (Add Item API, 5 points, partial), 1.2 (View Items API, 3 points, partial)
- **Tasks**:
  - **Task 1: Implement Item schema (2 points, 16 hours, Backend)**
    * Define schema with Mongoose (name, quantity, etc.) (8 hours).
    * Add text indexes for name, location; sparse index for sku (4 hours).
    * Write unit tests with Jest (4 hours).
  - **Task 2: Create POST /api/items endpoint (2 points, 16 hours, Backend)**
    * Implement route handler (6 hours).
    * Add validation for required fields (4 hours).
    * Write integration tests with Supertest (6 hours).
  - **Task 3: Create GET /api/items endpoint (1 point, 8 hours, Backend)**
    * Implement route handler (4 hours).
    * Write integration tests (4 hours).
- **Total Points**: 5
- **Total Hours**: 40 (Backend: 40, QA: 4 for test review)
- **Deliverable**: Functional backend API for adding/viewing items.
- **Assigned Roles**: Backend (2), QA (1)

Sprint 2 (Day 15–28): Web Frontend (Vue.js) Core

- **Goal**: Build Vue.js frontend for core CRUD.

- **Stories**: 1.1 (Add Item UI, 5 points, partial), 1.2 (View Items UI, 3 points), 1.3 (Edit Item UI, 5 points, partial), 1.4 (Delete Item UI, 3 points)
- **Tasks**:
  - **Task 1: Create HomePage.vue and ItemList.vue (2 points, 16 hours, Frontend)**
    - Set up Vue Router and HomePage.vue (6 hours).
    - Implement ItemList.vue with ItemCard.vue (6 hours).
    - Write unit tests with Vitest (4 hours).
  - **Task 2: Implement AddItemModal.vue (2 points, 16 hours, Frontend)**
    - Create modal with form (6 hours).
    - Integrate POST /api/items via Axios (6 hours).
    - Write E2E tests with Cypress (4 hours).
  - **Task 3: Implement EditItemModal.vue (2 points, 16 hours, Frontend)**
    - Create modal with pre-filled form (6 hours).
    - Integrate GET/PUT /api/items/:id (6 hours).
    - Write E2E tests (4 hours).
  - **Task 4: Add delete functionality (1 point, 8 hours, Frontend)**
    - Add delete button with prompt (4 hours).
    - Integrate DELETE /api/items/:id (4 hours).
- **Total Points**: 7
- **Total Hours**: 56 (Frontend: 56, QA: 8 for test review)
- **Deliverable**: Web app with basic CRUD.
- **Assigned Roles**: Frontend (2), QA (1)

Sprint 3 (Day 29–42): React Native Mobile Core

- **Goal**: Build React Native app for core CRUD.
- **Stories**: 1.1, 1.2, 1.3, 1.4 (Mobile UI)
- **Tasks**:
  - **Task 1: Create HomePage.js and ItemList.js (2 points, 16 hours, Swift/Android)**
    - Set up React Navigation and HomePage.js (6 hours).
    - Implement ItemList.js with FlatList (6 hours).
    - Write unit tests with Jest (4 hours).
  - **Task 2: Implement AddItem.js (2 points, 16 hours, Swift/Android)**
    - Create form screen (6 hours).
    - Integrate POST /api/items (6 hours).
    - Write E2E tests with Detox (4 hours).
  - **Task 3: Implement EditItem.js (2 points, 16 hours, Swift/Android)**
    - Create pre-filled form screen (6 hours).
    - Integrate GET/PUT /api/items/:id (6 hours).

        * Write E2E tests (4 hours).
    - **Task 4: Add delete functionality (1 point, 8 hours, Swift/Android)**
        * Add delete button with alert (4 hours).
        * Integrate DELETE /api/items/:id (4 hours).
- **Total Points**: 7
- **Total Hours**: 56 (Swift: 28, Android: 28, QA: 8 for test review)
- **Deliverable**: Mobile app with basic CRUD.
- **Assigned Roles**: Swift (1), Android (1), QA (1)

Sprint 4 (Day 43–56): Authentication

- **Goal**: Implement JWT-based authentication and RBAC.
- **Stories**: 2.1 (Login, 8 points), 2.2 (RBAC, 5 points)
- **Tasks**:
    - **Task 1: Create User schema and /api/auth endpoints (3 points, 24 hours, Backend)**
        * Define User schema with bcrypt (8 hours).
        * Implement /api/auth/login and /api/auth/register (8 hours).
        * Write tests (8 hours).
    - **Task 2: Implement login UI (3 points, 24 hours, Frontend/Swift/Android)**
        * Create Login.vue (8 hours, Frontend).
        * Create Login.js for React Native (8 hours, Swift/Android).
        * Integrate token storage (8 hours, Frontend/Swift/Android).
    - **Task 3: Add RBAC middleware and UI restrictions (2 points, 16 hours, Backend/Frontend)**
        * Implement auth middleware (8 hours, Backend).
        * Add role-based UI logic (8 hours, Frontend).
- **Total Points**: 8
- **Total Hours**: 64 (Backend: 24, Frontend: 16, Swift: 8, Android: 8, QA: 8 for test review)
- **Deliverable**: Secure app with login and roles.
- **Assigned Roles**: Backend (2), Frontend (1), Swift (1), Android (1), QA (1)

Sprint 5 (Day 57–70): Search and Filtering

- **Goal**: Add advanced search and filtering.
- **Stories**: 3.1 (Search/Filter, 5 points)
- **Tasks**:
    - **Task 1: Extend GET /api/items with query parameters (2 points, 16 hours, Backend)**
        * Add query parameter support (8 hours).
        * Write tests (8 hours).

  - **Task 2: Add search/filter UI (3 points, 24 hours, Frontend/Swift/Android)**
    * Implement search bar in HomePage.vue (8 hours, Frontend).
    * Implement search bar in HomePage.js (8 hours, Swift/Android).
    * Add filter dropdowns (8 hours, Frontend/Swift/Android).
- **Total Points**: 5
- **Total Hours**: 40 (Backend: 16, Frontend: 12, Swift: 6, Android: 6, QA: 4 for test review)
- **Deliverable**: Searchable inventory.
- **Assigned Roles**: Backend (1), Frontend (1), Swift (1), Android (1), QA (1)

Sprint 6 (Day 71–84): Offline Support

- **Goal**: Enable offline mode with sync.
- **Stories**: 3.2 (Offline Support, 8 points)
- **Tasks**:
  - **Task 1: Implement POST /api/sync endpoint (3 points, 24 hours, Backend)**
    * Create endpoint with conflict resolution (12 hours).
    * Write tests (12 hours).
  - **Task 2: Add local storage (3 points, 24 hours, Frontend/Swift/Android)**
    * Implement IndexedDB with dexie.js (8 hours, Frontend).
    * Implement AsyncStorage for React Native (8 hours, Swift/Android).
    * Write tests (8 hours, Frontend/Swift/Android).
  - **Task 3: Set up background sync (2 points, 16 hours, Swift/Android)**
    * Integrate react-native-background-fetch (8 hours).
    * Write tests (8 hours).
- **Total Points**: 8
- **Total Hours**: 64 (Backend: 24, Frontend: 12, Swift: 14, Android: 14, QA: 8 for test review)
- **Deliverable**: Offline-capable mobile app.
- **Assigned Roles**: Backend (2), Frontend (1), Swift (1), Android (1), QA (1)

Sprint 7 (Day 85–98): Real-Time Updates

- **Goal**: Add WebSocket-based real-time updates.
- **Stories**: 3.3 (WebSockets, 5 points)
- **Tasks**:
  - **Task 1: Integrate socket.io in backend (2 points, 16 hours, Backend)**
    * Set up socket.io server (8 hours).
    * Write tests (8 hours).

    - **Task 2: Add WebSocket listeners (3 points, 24 hours, Frontend/Swift/Android)**
        * Implement socket.io-client in Vue.js (8 hours, Frontend).
        * Implement socket.io-client in React Native (8 hours, Swift/Android).
        * Write tests (8 hours, Frontend/Swift/Android).
- **Total Points**: 5
- **Total Hours**: 40 (Backend: 16, Frontend: 12, Swift: 6, Android: 6, QA: 4 for test review)
- **Deliverable**: Real-time inventory updates.
- **Assigned Roles**: Backend (1), Frontend (1), Swift (1), Android (1), QA (1)

Sprint 8 (Day 99–112): Barcode Scanning

- **Goal**: Implement barcode/QR code scanning.
- **Stories**: 3.4 (Barcode Scanning, 5 points)
- **Tasks**:
    - **Task 1: Add GET /api/items/by-sku endpoint (2 points, 16 hours, Backend)**
        * Implement endpoint (8 hours).
        * Write tests (8 hours).
    - **Task 2: Implement scanning UI (3 points, 24 hours, Frontend/Swift/Android)**
        * Add getUserMedia scanning in Vue.js (8 hours, Frontend).
        * Add vision-camera scanning in React Native (8 hours, Swift/Android).
        * Write tests (8 hours, Frontend/Swift/Android).
- **Total Points**: 5
- **Total Hours**: 40 (Backend: 16, Frontend: 12, Swift: 6, Android: 6, QA: 4 for test review)
- **Deliverable**: Barcode scanning feature.
- **Assigned Roles**: Backend (1), Frontend (1), Swift (1), Android (1), QA (1)

Sprint 9 (Day 113–126): Performance Optimization

- **Goal**: Optimize performance with pagination and caching.
- **Stories**: 4.1 (Performance, 5 points)
- **Tasks**:
    - **Task 1: Add pagination to GET /api/items (2 points, 16 hours, Backend)**
        * Implement pagination (8 hours).
        * Write tests (8 hours).
    - **Task 2: Integrate Redis caching (2 points, 16 hours, Backend)**
        * Set up Redis client (8 hours).

        * Write tests (8 hours).
    - **Task 3: Optimize frontend lists (1 point, 8 hours, Frontend)**
        * Add virtual scrolling in ItemList.vue (4 hours).
        * Optimize FlatList in React Native (4 hours).
- **Total Points**: 5
- **Total Hours**: 40 (Backend: 24, Frontend: 12, QA: 4 for test review)
- **Deliverable**: Faster app performance.
- **Assigned Roles**: Backend (2), Frontend (1), QA (1)

Sprint 10 (Day 127–140): Analytics and CI/CD

- **Goal**: Add analytics dashboard and CI/CD pipeline.
- **Stories**: 3.5 (Analytics, 5 points), 6.3 (CI/CD, 5 points)
- **Tasks**:
    - **Task 1: Create /api/analytics endpoints (2 points, 16 hours, Backend)**
        * Implement summary and low-stock endpoints (8 hours).
        * Write tests (8 hours).
    - **Task 2: Build dashboard UI (3 points, 24 hours, Frontend)**
        * Create Dashboard.vue with Chart.js (12 hours).
        * Create Dashboard.js with react-native-chart-kit (12 hours).
    - **Task 3: Set up CI/CD pipeline (3 points, 24 hours, DevOps)**
        * Configure GitHub Actions for Jest, Vitest, Cypress, Detox (12 hours).
        * Set up deployment to Netlify and AWS (12 hours).
- **Total Points**: 8
- **Total Hours**: 64 (Backend: 16, Frontend: 24, DevOps: 24, QA: 8 for test review)
- **Deliverable**: Analytics dashboard, automated testing/deployment.
- **Assigned Roles**: Backend (1), Frontend (2), DevOps (1), QA (1)

Sprint 11 (Day 141–154): Native Mobile (Swift iOS)

- **Goal**: Prototype native iOS app with SwiftUI.
- **Stories**: 6.1 (Swift iOS, 8 points)
- **Tasks**:
    - **Task 1: Build SwiftUI app (5 points, 40 hours, Swift)**
        * Create HomeView with NavigationStack (12 hours).
        * Implement AddItemView and EditItemView (16 hours).
        * Add ViewItemView (12 hours).
    - **Task 2: Integrate with API (3 points, 24 hours, Swift)**
        * Set up URLSession for CRUD operations (12 hours).
        * Write tests with XCTest (12 hours).
- **Total Points**: 8
- **Total Hours**: 64 (Swift: 64, QA: 8 for test review)
- **Deliverable**: Native iOS app prototype.

- **Assigned Roles**: Swift (2), QA (1)

Sprint 12 (Day 155–168): Native Mobile (Kotlin Android), Accessibility, IoT

- **Goal**: Prototype native Android app, add accessibility and IoT.
- **Stories**: 5.1 (IoT, 8 points), 5.2 (Accessibility, 5 points), 6.2 (Kotlin Android, 8 points)
- **Tasks**:
  - **Task 1: Build Jetpack Compose app (5 points, 40 hours, Android)**
    - Create HomeScreen with NavHost (12 hours).
    - Implement AddItemScreen and EditItemScreen (16 hours).
    - Add ViewItemScreen (12 hours).
  - **Task 2: Implement IoT endpoint and RFID support (3 points, 24 hours, Backend/Android)**
    - Create POST /api/iot/rfid endpoint (12 hours, Backend).
    - Add RFID integration for Android (12 hours, Android).
  - **Task 3: Add accessibility features (2 points, 16 hours, Frontend/Swift/Android)**
    - Add ARIA labels in Vue.js (6 hours, Frontend).
    - Add accessibilityLabel in React Native and native apps (10 hours, Swift/Android).
- **Total Points**: 10
- **Total Hours**: 80 (Backend: 12, Frontend: 6, Swift: 5, Android: 57, QA: 8 for test review)
- **Deliverable**: Native Android app prototype, accessible UI, IoT integration.
- **Assigned Roles**: Backend (1), Frontend (1), Swift (1), Android (2), QA (1)

Post-Sprint (Day 169–181): Final Testing and Deployment

- **Tasks**:
  - **Task 1: End-to-end testing (24 hours, QA/Swift/Android)**
    - Test web, React Native, Swift iOS, Kotlin Android (12 hours, QA).
    - Validate native app integration (12 hours, Swift/Android).
  - **Task 2: Deploy to production (16 hours, DevOps)**
    - Deploy web to Netlify (4 hours).
    - Deploy backend to AWS (8 hours).
    - Publish mobile apps to App Store/Google Play (4 hours).
  - **Task 3: Fix critical bugs (16 hours, Frontend/Swift/Android)**
    - Address UI/UX issues (8 hours, Frontend).
    - Fix native app bugs (8 hours, Swift/Android).
- **Total Hours**: 56 (Frontend: 8, Swift: 10, Android: 10, QA: 12, DevOps:

16)
- **Deliverable**: Production-ready Inventory App.
- **Assigned Roles**: Frontend (1), Swift (1), Android (1), QA (1), DevOps (1)

Gantt Chart

The Gantt chart visualizes the sprint timeline, tasks, and dependencies using Mermaid syntax. Each sprint is 14 days, with the post-sprint phase lasting 13 days.

```
gantt
  title Inventory App Development Timeline
  dateFormat MM-DD
  axisFormat Week %U
  todayMarker off

  section Sprint 1 -<br/>Backend & Database
    1.1: Item Schema            :s1a, 01-01, 6d
    1.2: POST /api/items        :s1b, 01-07, 5d
    1.3: GET /api/items         :s1c, 01-12, 3d

  section Sprint 2 -<br/>Web Frontend
    2.1: HomePage/ItemList      :s2a, 01-15, 5d
    2.2: AddItemModal           :s2b, 01-20, 5d
    2.3: EditItemModal          :s2c, 01-25, 4d
    2.4: Delete Functionality   :s2d, 01-25, 2d

  section Sprint 3 -<br/>React Native Core
    3.1: HomePage/ItemList      :s3a, 01-29, 5d
    3.2: AddItem Screen         :s3b, 02-03, 5d
    3.3: EditItem Screen        :s3c, 02-08, 4d
    3.4: Delete Functionality   :s3d, 02-08, 2d

  section Sprint 4 -<br/>Authentication
    4.1: User Schema/Endpoints  :s4a, 02-12, 6d
    4.2: Login UI               :s4b, 02-18, 5d
    4.3: RBAC Middleware/UI     :s4c, 02-23, 3d

  section Sprint 5 -<br/>Search & Filtering
    5.1: Query Parameters       :s5a, 02-26, 5d
    5.2: Search/Filter UI       :s5b, 03-02, 9d

  section Sprint 6 -<br/>Offline Support
    6.1: Sync Endpoint          :s6a, 03-11, 6d
    6.2: Local Storage          :s6b, 03-17, 5d
```

```
  6.3: Background Sync        :s6c, 03-22, 3d

section Sprint 7 -<br/>Real-Time Updates
  7.1: Socket.io Backend      :s7a, 03-25, 5d
  7.2: WebSocket Listeners    :s7b, 03-30, 9d

section Sprint 8 -<br/>Barcode Scanning
  8.1: By-SKU Endpoint        :s8a, 04-08, 5d
  8.2: Scanning UI            :s8b, 04-13, 9d

section Sprint 9 -<br/>Performance Optimization
  9.1: Pagination             :s9a, 04-22, 5d
  9.2: Redis Caching          :s9b, 04-27, 5d
  9.3: Frontend Optimization  :s9c, 05-02, 4d

section Sprint 10 -<br/>Analytics & CI/CD
  10.1: Analytics Endpoints    :s10a, 05-06, 5d
  10.2: Dashboard UI           :s10b, 05-11, 5d
  10.3: CI/CD Pipeline         :s10c, 05-16, 4d

section Sprint 11 -<br/>Native iOS (Swift)
  11.1: SwiftUI App            :s11a, 05-20, 9d
  11.2: API Integration        :s11b, 05-29, 5d

section Sprint 12 -<br/>Native Android,<br/>Accessibility,<br/>IoT
  12.1: Jetpack Compose App    :s12a, 06-03, 5d
  12.2: IoT Endpoint/RFID      :s12b, 06-08, 5d
  12.3: Accessibility Features :s12c, 06-13, 4d

section Post-Sprint -<br/>Testing & Deployment
  E2E Testing             :post1, 06-17, 5d
  Deployment              :post2, 06-22, 4d
  Bug Fixes               :post3, 06-26, 4d
```

Milestone Table

Milestone	Description	Completion Day	Stories Completed
M1: Backend Foundation	Functional API and MongoDB schema for core CRUD	Day 14	1.5, 1.1 (partial), 1.2 (partial)
M2: Web MVP	Vue.js app with core CRUD	Day 28	1.1, 1.2, 1.3, 1.4
M3: Mobile MVP	React Native app with core CRUD	Day 42	1.1, 1.2, 1.3, 1.4

Milestone	Description	Completion Day	Stories Completed
M4: Secure MVP	Authentication and RBAC implemented	Day 56	2.1, 2.2
M5: Enhanced Features	Search, offline support, WebSockets, barcode scanning	Day 112	3.1, 3.2, 3.3, 3.4
M6: Optimized App	Performance enhancements with pagination and caching	Day 126	4.1
M7: Analytics & CI/CD	Analytics dashboard and automated testing/deployment	Day 140	3.5, 6.3
M8: Native Mobile Prototypes	Swift iOS and Kotlin Android apps, accessibility, IoT	Day 168	5.1, 5.2, 6.1, 6.2
M9: Production Release	Fully tested, deployed app across all platforms	Day 181	All stories

Dependencies and Risks

Dependencies

- **Backend API** (Sprint 1, Day 1–14) must precede frontend development (Sprints 2–3, Day 15–42).
- **Authentication** (Sprint 4, Day 43–56) is required for RBAC-dependent features (e.g., analytics in Sprint 10, Day 127–140).
- **CI/CD Pipeline** (Sprint 10, Day 127–140) enhances testing for native apps (Sprints 11–12, Day 141–168).
- **Search and Filtering** (Sprint 5, Day 57–70) supports analytics data (Sprint 10, Day 127–140).

Risks

- **Team Skill Gaps**: Limited Swift/Kotlin expertise. **Mitigation**: Cross-train in Sprints 1–4 (Day 1–56).
- **Integration Complexity**: WebSockets/IoT compatibility issues. **Mitigation**: Prototype in Sprint 7 (Day 85–98) and Sprint 12 (Day 155–168).
- **Scope Creep**: Could-have features (analytics, IoT) may delay core features. **Mitigation**: Adhere to MoSCoW prioritization.
- **Performance Bottlenecks**: Large inventories may slow app. **Mitigation**: Prioritize performance in Sprint 9 (Day 113–126).

Resource Allocation

- **Frontend Developers (2)**: Sprints 2, 5, 7–10, 12 (Day 15–28, 57–140, 155–168) for Vue.js and accessibility.
- **Backend Developers (2)**: Sprints 1, 4–10, 12 (Day 1–14, 43–140, 155–168) for Express.js, MongoDB, IoT.
- **Swift iOS Developers (2)**: Sprints 3–8 (Day 29–112) for React Native, Sprint 11–12 (Day 141–168) for SwiftUI.
- **Android Developers (2)**: Sprints 3–8 (Day 29–112) for React Native, Sprint 12 (Day 155–168) for Jetpack Compose.
- **QA Engineer (1)**: Sprints 1–12 (Day 1–168) for testing, Post-Sprint (Day 169–181) for E2E testing.
- **DevOps Engineer (1)**: Sprint 10 (Day 127–140) for CI/CD, Post-Sprint (Day 169–181) for deployment.

Metrics and Success Criteria

Velocity

- Target 5–8 points per sprint per developer pair, adjusting based on complexity.

Definition of Done

- Code reviewed, merged into main.
- Tests pass (90% coverage).
- Deployed to staging, verified by QA.
- Documentation updated.

Success Criteria

- **Day 42 (Sprint 3)**: Core CRUD for web and React Native.
- **Day 84 (Sprint 6)**: Must-have enhancements (authentication, search, offline).
- **Day 168 (Sprint 12)**: Native iOS/Android prototypes, Should-have/Could-have enhancements.
- **Day 181 (Post-Sprint)**: Production-ready app, zero critical bugs.

Instructions for Use

- **Import into Agile Tools**: Convert backlog to Jira/Trello/Azure DevOps tasks, assigning stories to sprints (e.g., Sprint 1 = Day 1–14).
- **Customize Timeline**: Map project days to your start date (e.g., Day 1 = project kickoff). 181 days 36 weeks (9 months) with a 5-day workweek.
- **Track Progress**: Use sprint plan for planning sessions, ensuring dependencies (e.g., backend before frontend) are met.

Additional Notes

- **Prioritization**: Must-have features (CRUD, authentication) are front-loaded (Day 1–56) for an early MVP. Should-have enhancements (search, offline, WebSockets, barcode, performance, accessibility) span Day 57–126. Could-have features (analytics, IoT, native apps) are in Day 127–168 for flexibility.
- **Team Specialization**: 2 Swift and 2 Android developers ensure native app expertise, with cross-training in Day 1–56 to mitigate skill gaps.
- **Stakeholder Engagement**: Sprint reviews validate features, especially native apps (Day 141–168).
- **Risk Mitigation**: Early prototyping (WebSockets: Day 85–98, IoT: Day 155–168) and strict prioritization reduce risks.

Product Owner: *TBD*
Scrum Master: *TBD*
Date Prepared: 2025-05-15