pharmy_jira_complete_workflow

PHARMY PROJECT - COMPLETE JIRA WORKFLOW GUIDE

PROJECT OVERVIEW

Project Name: Pharmy - Medicine Location Platform
Start Date: November 29, 2024
End Date: January 1, 2025
Total Duration: 34 days across 4 sprints
Team Size: 5 members

EPICS SUMMARY

Epic Key	Epic Name	Story Points	Description
PHARMY-1	Project Foundation & Setup	13	Development environment setup, project structure, database schema, and design finalization
PHARMY-2	Authentication System	13	User registration, login, JWT authentication, and protected routes for all three user roles
PHARMY-3	Map Interface & Location Services	13	Interactive map display, user location detection, pharmacy markers, and map-based navigation
PHARMY-4	Medicine Search System	13	Medicine search with autocomplete, map integration, distance calculation, and results display
PHARMY-5	Pharmacy Management	8	Pharmacy registration, admin verification, and profile management
PHARMY-6	Inventory Management	13	Inventory CRUD operations, CSV bulk upload, and inventory history tracking
PHARMY-7	Reports & Admin System	8	User reporting system for inaccuracies and admin dashboard for report resolution

Epic Key	Epic Name	Story Points	Description
PHARMY-8	Testing, Polish & Deployment	13	Bug fixes, UI/UX polish, performance optimization, deployment, and demo preparation

---

JIRA LABELS LIBRARY

Technology Stack Labels

- `frontend` - Next.js, React, UI work
- `backend` - Laravel, PHP, API work
- `database` - PostgreSQL, migrations, queries
- `integration` - Connecting frontend to backend
- `devops` - Server, hosting, CI/CD

Feature Domain Labels

- `authentication` - Login, registration, JWT, sessions
- `map` - Google Maps/Mapbox integration
- `search` - Medicine search functionality
- `inventory` - Inventory management features
- `reports` - Reporting system
- `admin` - Admin dashboard and features
- `pharmacy` - Pharmacy-specific features
- `user` - General user features

Work Type Labels

- `setup` - Initial configuration and installation
- `design` - UI/UX design work
- `documentation` - Docs, guides, README
- `testing` - QA, unit tests, integration tests
- `bug` - Bug fixes
- `refactor` - Code improvement without new features
- `optimization` - Performance improvements

Priority & Status Labels

- `urgent` - Critical, blocking other work

- `high-priority` - Important, should do soon
- `low-priority` - Can wait if needed
- `blocked` - Cannot proceed until dependency resolved
- `must-have` - Core MVP feature
- `should-have` - Important but can cut if behind
- `could-have` - Nice to have, cut if needed

Platform Labels

- `mobile` - Mobile-specific work
- `responsive` - Responsive design
- `cross-browser` - Browser compatibility
- `ui` - User interface work

Sprint Labels

- `sprint-0` - Foundation setup
- `sprint-1` - Auth + Map
- `sprint-2` - Search
- `sprint-3` - Pharmacy + Inventory
- `sprint-4` - Reports + Polish
- `deployment` - Deployment phase
- `presentation` - Final presentation prep

Special Labels

- `pair-programming` - Best done with 2+ people
- `ai-assisted` - Good candidate for AI help
- `learning` - New concept for team
- `security` - Security-related work
- `performance` - Performance-critical
- `accessibility` - A11y improvements
- `qa` - Quality assurance

SPRINT BREAKDOWN

SPRINT 0: FOUNDATION SETUP

Duration: November 29 - December 5, 2024 (7 days)
Goal: Get everyone's environment ready and design finalized

STORY 1: Development Environment Setup

Sprint: Sprint 0 (November 29 - December 5, 2024)
Epic: Project Foundation & Setup (PHARMY-1)
Story Points: 6
Labels: `setup`, `sprint-0`, `must-have`, `learning`

User Story:
As a developer
I want to set up my development environment
So that I can start coding the project

Acceptance Criteria:

- ✅ Node.js, pnpm, PostgreSQL, PHP, Composer, Laravel installed
- ✅ Antigravity IDE configured
- ✅ Git repository created with branching strategy
- ✅ All team members can access the repository

Tasks:

Task 1.1: Install development tools (Node.js, pnpm, PostgreSQL, PHP, Composer, Laravel)

- **Story Points:** 2
- **Labels:** `setup`, `sprint-0`, `must-have`, `learning`
- **Description:** Install all required development tools on each team member's machine. Verify installations with version checks.

Task 1.2: Setup Git repository and branching strategy

- **Story Points:** 2
- **Labels:** `setup`, `devops`, `sprint-0`, `must-have`
- **Description:** Create Git repository, define branching strategy (main, develop, feature branches), set up branch protection rules.

Task 1.3: Create Jira board and project workflow

- **Story Points:** 1
- **Labels:** `setup`, `sprint-0`, `must-have`, `documentation`

- **Description:** Set up Jira Scrum board, create workflow states (To Do, In Progress, Code Review, Testing, Done), add labels.

Task 1.4: Setup communication channels (Google Meet, Discord/Slack)

- **Story Points:** 1
- **Labels:** `setup`, `sprint-0`, `must-have`
- **Description:** Create Google Meet recurring standup link, set up Discord/Slack workspace with channels for general, backend, frontend, blockers.

STORY 2: Project Structure Creation

Sprint: Sprint 0 (November 29 - December 5, 2024)
Epic: Project Foundation & Setup (PHARMY-1)
Story Points: 18
Labels: `frontend`, `backend`, `database`, `setup`, `sprint-0`, `must-have`

User Story:
As a developer
I want to have a working project structure
So that I can start building features

Acceptance Criteria:

- ✅ Next.js project created with TypeScript + shadcn/ui
- ✅ Laravel API project created
- ✅ PostgreSQL database setup complete
- ✅ All 6 database tables implemented and migrated
- ✅ Initial seed data loaded (1 admin, 20-30 medicines)
- ✅ Frontend can call backend test endpoint

Tasks:

Task 2.1: Create Next.js project with TypeScript and shadcn/ui

- **Story Points:** 3
- **Labels:** `frontend`, `setup`, `sprint-0`, `must-have`
- **Description:** Initialize Next.js 14+ project with TypeScript, install shadcn/ui, configure Tailwind CSS, set up folder structure (components, app, lib, types).

Task 2.2: Create Laravel API project structure

- **Story Points:** 3

- **Labels:** `backend`, `setup`, `sprint-0`, `must-have`
- **Description:** Initialize Laravel project, configure .env file, set up routes/api.php, create base controllers, install required packages (JWT, CORS).

Task 2.3: Setup PostgreSQL database and connection

- **Story Points:** 2
- **Labels:** `backend`, `database`, `setup`, `sprint-0`, `must-have`
- **Description:** Create PostgreSQL database, configure Laravel database connection, test connection, set up pgAdmin for database management.

Task 2.4: Implement database schema (6 tables) and migrations

- **Story Points:** 5
- **Labels:** `backend`, `database`, `sprint-0`, `must-have`, `pair-programming`
- **Description:** Create migrations for: users, pharmacies, medicines, inventory, reports, inventory_history tables. Include all fields, foreign keys, indexes.

Task 2.5: Create database seeds (admin user, 20-30 medicines)

- **Story Points:** 2
- **Labels:** `backend`, `database`, `setup`, `sprint-0`, `must-have`
- **Description:** Create seeders for 1 admin user, 20-30 realistic medicine entries with categories. Run seeders and verify data.

Task 2.6: Create test API endpoint and connect Next.js

- **Story Points:** 3
- **Labels:** `backend`, `frontend`, `integration`, `setup`, `sprint-0`, `must-have`
- **Description:** Create /api/test endpoint in Laravel, call it from Next.js, verify CORS configuration, ensure data flows correctly.

STORY 3: Design & Documentation

Sprint: Sprint 0 (November 29 - December 5, 2024)
Epic: Project Foundation & Setup (PHARMY-1)
Story Points: 9
Labels: `design`, `documentation`, `sprint-0`, `must-have`

User Story:
As a team member
I want to have clear design and API documentation
So that I can build features consistently

Acceptance Criteria:

- ✅ Figma designs reviewed and finalized
- ✅ Component specifications documented
- ✅ API contract fully defined (all endpoints, request/response)
- ✅ Setup instructions documented

Tasks:

Task 3.1: Review and finalize Figma designs

- **Story Points:** 2
- **Labels:** `design`, `documentation`, `sprint-0`, `must-have`
- **Description:** Review all Figma screens, finalize color scheme, typography, spacing standards. Get team consensus on design decisions.

Task 3.2: Document component specifications

- **Story Points:** 2
- **Labels:** `design`, `documentation`, `sprint-0`, `frontend`, `must-have`
- **Description:** Document all reusable components (buttons, forms, cards, modals), their props, variants, and usage examples.

Task 3.3: Define complete API contract

- **Story Points:** 3
- **Labels:** `backend`, `documentation`, `sprint-0`, `must-have`, `pair-programming`
- **Description:** List ALL API endpoints for the project with HTTP methods, request bodies, response formats, error codes. This is the master contract.

Task 3.4: Create setup and API documentation

- **Story Points:** 2
- **Labels:** `documentation`, `sprint-0`, `must-have`
- **Description:** Write README with setup instructions, environment variables needed, how to run migrations/seeds, how to start both servers.

SPRINT 1: AUTHENTICATION + BASIC MAP

Duration: December 6 - December 12, 2024 (7 days)
Goal: Users can login/register + Map displays with user location

---

STORY 4: User Registration

Sprint: Sprint 1 (December 6 - December 12, 2024)
Epic: Authentication System (PHARMY-2)
Story Points: 14
Labels: `backend`, `frontend`, `authentication`, `security`, `sprint-1`, `must-have`

User Story:
As a new user
I want to create an account
So that I can access the platform

Acceptance Criteria:

- ✅ User can register with name, email, password, role
- ✅ Email validation works
- ✅ Passwords are securely hashed
- ✅ JWT token returned on success
- ✅ Error messages display for invalid inputs
- ✅ Redirect to login page after successful registration

Tasks:

Task 4.1: Build user registration API endpoint with validation

- **Story Points:** 5
- **Labels:** `backend`, `authentication`, `security`, `sprint-1`, `must-have`, `ai-assisted`
- **Description:** Create POST /api/register endpoint. Validate email format, password strength. Hash password with bcrypt. Assign role (user/pharmacy/admin). Return JWT token and user data.

Task 4.2: Create registration page UI with form and validation

- **Story Points:** 5
- **Labels:** `frontend`, `authentication`, `sprint-1`, `must-have`, `ui`
- **Description:** Create /register page with form fields (name, email, password, confirm password, role select). Add client-side validation, error display, loading state.

Task 4.3: Connect registration form to backend API

- **Story Points:** 2
- **Labels:** `frontend`, `backend`, `integration`, `authentication`, `sprint-1`, `must-have`

- **Description:** Connect form submission to API, handle responses, store JWT token, display errors, redirect to login on success.

Task 4.4: Test registration flow end-to-end

- **Story Points:** 2
- **Labels:** `testing`, `authentication`, `sprint-1`, `must-have`
- **Description:** Test all three roles registration, test validation errors, test duplicate email, verify JWT token generation, test redirect flow.

STORY 5: User Login

Sprint: Sprint 1 (December 6 - December 12, 2024)
Epic: Authentication System (PHARMY-2)
Story Points: 18
Labels: `backend`, `frontend`, `authentication`, `security`, `sprint-1`, `must-have`

User Story:
As a registered user
I want to login to my account
So that I can use the platform

Acceptance Criteria:

- ✅ User can login with email and password
- ✅ Valid credentials return JWT token
- ✅ Invalid credentials show error message
- ✅ User redirected to map page after login
- ✅ User data stored securely

Tasks:

Task 5.1: Build login API endpoint with JWT generation

- **Story Points:** 3
- **Labels:** `backend`, `authentication`, `security`, `sprint-1`, `must-have`, `ai-assisted`
- **Description:** Create POST /api/login endpoint. Validate credentials against database. Generate JWT token with user ID, role, expiration. Return token and user data.

Task 5.2: Create authentication middleware for protected routes

- **Story Points:** 3
- **Labels:** `backend`, `authentication`, `security`, `sprint-1`, `must-have`, `pair-programming`
- **Description:** Create middleware to verify JWT tokens. Check token validity, expiration. Attach user data to request. Return 401 for invalid tokens.

Task 5.3: Create login page UI with form

- **Story Points:** 3
- **Labels:** `frontend`, `authentication`, `sprint-1`, `must-have`, `ui`
- **Description:** Create /login page with email and password fields. Add "Remember me" checkbox, "Forgot password" link (UI only), error display, loading state.

Task 5.4: Implement auth context and secure token storage

- **Story Points:** 5
- **Labels:** `frontend`, `authentication`, `security`, `sprint-1`, `must-have`, `pair-programming`
- **Description:** Create React Context for auth state. Store JWT token securely (httpOnly cookie or secure alternative to localStorage). Provide login/logout/getUser functions.

Task 5.5: Create protected route wrapper

- **Story Points:** 2
- **Labels:** `frontend`, `authentication`, `security`, `sprint-1`, `must-have`
- **Description:** Create HOC or component to protect routes. Check if user is authenticated. Redirect to login if not. Allow access based on role.

Task 5.6: Connect login form to backend and test flow

- **Story Points:** 2
- **Labels:** `frontend`, `backend`, `integration`, `authentication`, `sprint-1`, `must-have`
- **Description:** Connect form to API, handle success/error responses, store token, update auth context, redirect to /map, test complete flow.

STORY 6: Map Interface Display

Sprint: Sprint 1 (December 6 - December 12, 2024)
Epic: Map Interface & Location Services (PHARMY-3)
Story Points: 26
Labels: `frontend`, `backend`, `map`, `sprint-1`, `must-have`, `high-priority`

User Story:
As a user
I want to see an interactive map
So that I can find pharmacies visually

Acceptance Criteria:

- ✅ Full-screen map displays as main interface
- ✅ Search bar overlay visible at top
- ✅ User's current location shows on map
- ✅ Map centers on user location (or Beirut if denied)
- ✅ All verified pharmacies show as markers
- ✅ Clicking marker shows pharmacy info popup
- ✅ Map styling matches prototype
- ✅ Map is responsive on mobile

Tasks:

Task 6.1: Setup map API (Google Maps/Mapbox) and get API key

- **Story Points:** 2
- **Labels:** `frontend`, `map`, `setup`, `sprint-1`, `must-have`
- **Description:** Choose between Google Maps or Mapbox. Create account, get API key, install npm packages, configure API key in environment variables.

Task 6.2: Create main map component with full-screen layout

- **Story Points:** 5
- **Labels:** `frontend`, `map`, `sprint-1`, `must-have`, `high-priority`, `pair-programming`
- **Description:** Create /map page with full-screen map component. Set default center (Beirut: 33.8938° N, 35.5018° E). Set default zoom level. Add map controls.

Task 6.3: Implement user location detection and marker

- **Story Points:** 3
- **Labels:** `frontend`, `map`, `sprint-1`, `must-have`, `learning`
- **Description:** Request geolocation permission. Get user's current coordinates. Display user location marker (custom icon). Handle permission denied gracefully.

Task 6.4: Create search bar overlay UI (non-functional)

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `map`, `sprint-1`, `must-have`

- **Description:** Create search bar component overlaying top of map. Style to match prototype. Add search icon, input field. Make responsive. Will be functional in Sprint 2.

Task 6.5: Build API endpoint to fetch verified pharmacies

- **Story Points:** 3
- **Labels:** `backend`, `pharmacy`, `database`, `sprint-1`, `must-have`
- **Description:** Create GET /api/pharmacies endpoint. Return only verified pharmacies. Include: id, name, address, latitude, longitude, phone. Optimize query.

Task 6.6: Display pharmacy markers on map with custom icons

- **Story Points:** 5
- **Labels:** `frontend`, `map`, `sprint-1`, `must-have`, `design`
- **Description:** Fetch pharmacies from API. Create marker for each pharmacy. Use custom marker icons (pharmacy cross icon). Cluster markers if too many in same area.

Task 6.7: Create info popup for marker clicks

- **Story Points:** 3
- **Labels:** `frontend`, `map`, `ui`, `sprint-1`, `must-have`
- **Description:** Create popup/info window component. Show on marker click. Display: pharmacy name, address, phone. Style to match prototype. Add close button.

Task 6.8: Connect map to pharmacies API and test

- **Story Points:** 3
- **Labels:** `frontend`, `backend`, `integration`, `map`, `sprint-1`, `must-have`
- **Description:** Integrate all pieces. Test map loads, user location works, pharmacies display, popups work. Test on desktop and mobile. Verify responsiveness.

SPRINT 2: MEDICINE SEARCH ON MAP

Duration: December 13 - December 19, 2024 (7 days)
Goal: Users can search for medicines and see available pharmacies highlighted on the map

---

STORY 7: Medicine Search with Autocomplete

Sprint: Sprint 2 (December 13 - December 19, 2024)
Epic: Medicine Search System (PHARMY-4)
Story Points: 21
Labels: `backend`, `frontend`, `search`, `database`, `sprint-2`, `must-have`

User Story:
As a user
I want to search for medicines by name
So that I can find pharmacies that have them

Acceptance Criteria:

- ✅ Search bar shows autocomplete suggestions as I type
- ✅ Search returns pharmacies with the medicine available
- ✅ Results include distance from my location
- ✅ Results sorted by distance
- ✅ Empty state displays when no results found
- ✅ Loading indicator shows during search

Tasks:

Task 7.1: Create medicine search API with partial match

- **Story Points:** 5
- **Labels:** `backend`, `search`, `database`, `sprint-2`, `must-have`, `ai-assisted`
- **Description:** Create POST /api/search endpoint. Accept medicine name. Use LIKE query for partial match. Join inventory table. Return pharmacies with medicine available. Include pharmacy details.

Task 7.2: Implement distance calculation from user location

- **Story Points:** 5
- **Labels:** `backend`, `search`, `sprint-2`, `must-have`, `learning`, `pair-programming`
- **Description:** Accept user lat/long in request. Calculate distance using Haversine formula or PostGIS. Return distance in kilometers. Sort results by distance ascending.

Task 7.3: Add database indexes for search optimization

- **Story Points:** 2
- **Labels:** `backend`, `database`, `optimization`, `performance`, `sprint-2`
- **Description:** Add index on medicines.name for faster LIKE queries. Add index on inventory.medicine_id and pharmacy_id. Test query performance before/after.

Task 7.4: Make search bar functional with autocomplete

- **Story Points:** 5
- **Labels:** `frontend`, `search`, `ui`, `sprint-2`, `must-have`
- **Description:** Add onChange handler to search input. Implement debounced search (300ms). Show autocomplete dropdown with medicine suggestions. Handle keyboard navigation (arrow keys, enter).

Task 7.5: Add loading states and error handling

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `sprint-2`, `must-have`
- **Description:** Add loading spinner during search. Show error message if API fails. Add empty state "No results found" message. Add "Try another search" suggestion.

Task 7.6: Connect search to backend API and test

- **Story Points:** 2
- **Labels:** `frontend`, `backend`, `integration`, `search`, `sprint-2`, `must-have`
- **Description:** Connect search input to API endpoint. Test with various medicine names. Verify distance calculation. Test edge cases (no results, API errors, slow network).

STORY 8: Map Integration with Search Results

Sprint: Sprint 2 (December 13 - December 19, 2024)
Epic: Medicine Search System (PHARMY-4)
Story Points: 22
Labels: `frontend`, `map`, `search`, `ui`, `sprint-2`, `must-have`

User Story:
As a user
I want to see search results highlighted on the map
So that I can visually locate pharmacies with my medicine

Acceptance Criteria:

- ✅ Pharmacies with searched medicine highlighted differently
- ✅ Different marker colors for available vs not available
- ✅ Results sidebar shows pharmacy list with distances
- ✅ Clicking sidebar item centers map on that pharmacy
- ✅ Enhanced popup shows medicine availability details
- ✅ "Clear search" button resets map
- ✅ Last updated timestamp visible

Tasks:

Task 8.1: Implement marker highlighting based on search

- **Story Points:** 5
- **Labels:** `frontend`, `map`, `search`, `sprint-2`, `must-have`, `design`
- **Description:** Change marker colors based on search results. Green = has medicine available. Gray = doesn't have medicine. Scale up markers with searched medicine. Dim others.

Task 8.2: Create results sidebar/panel with pharmacy list

- **Story Points:** 5
- **Labels:** `frontend`, `ui`, `search`, `sprint-2`, `must-have`
- **Description:** Create sidebar showing list of pharmacies with searched medicine. Display: name, address, distance, last updated. Make scrollable. Show "X results found" header.

Task 8.3: Enhance info popup with medicine availability details

- **Story Points:** 3
- **Labels:** `frontend`, `map`, `ui`, `sprint-2`, `must-have`
- **Description:** Update popup to show: pharmacy info, medicine availability status (Available/Not Available), last updated timestamp, distance from user, notes (if any).

Task 8.4: Add "Clear search" functionality

- **Story Points:** 2
- **Labels:** `frontend`, `search`, `ui`, `sprint-2`, `must-have`
- **Description:** Add "Clear" or "X" button to search bar. Reset map to show all pharmacies. Clear search input. Close sidebar. Reset marker colors.

Task 8.5: Add empty state for no results

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `search`, `sprint-2`, `must-have`
- **Description:** Show message when search returns no pharmacies. Suggest: "Try another medicine" or "No pharmacies have this medicine yet". Keep search bar visible.

Task 8.6: Test complete search flow and edge cases

- **Story Points:** 3
- **Labels:** `testing`, `search`, `sprint-2`, `must-have`, `qa`

- **Description:** Test: search with results, search with no results, clear search, click markers, click sidebar items, mobile responsiveness, slow network, API errors.

SPRINT 3: PHARMACY REGISTRATION + INVENTORY MANAGEMENT

Duration: December 20 - December 26, 2024 (7 days)
Goal: Pharmacies can register and manage their inventory

STORY 9: Pharmacy Registration & Verification

Sprint: Sprint 3 (December 20 - December 26, 2024)
Epic: Pharmacy Management (PHARMY-5)
Story Points: 21
Labels: `backend`, `frontend`, `pharmacy`, `admin`, `sprint-3`, `must-have`

User Story:
As a pharmacy owner
I want to register my pharmacy
So that I can appear on the map after admin approval

Acceptance Criteria:

- ✅ Pharmacy can register with business details
- ✅ Registration includes location coordinates
- ✅ Pharmacy status starts as "pending"
- ✅ Admin can view pending registrations
- ✅ Admin can approve/reject pharmacies
- ✅ Approved pharmacies appear on map

Tasks:

Task 9.1: Build pharmacy registration API endpoint

- **Story Points:** 5
- **Labels:** `backend`, `pharmacy`, `sprint-3`, `must-have`, `ai-assisted`
- **Description:** Create POST /api/pharmacy/register endpoint. Accept: pharmacy name, address, lat/long, phone, license number. Set verified=false. Create pharmacy user account. Return success message.

Task 9.2: Create pharmacy registration form UI

- **Story Points:** 5
- **Labels:** `frontend`, `pharmacy`, `sprint-3`, `must-have`, `ui`
- **Description:** Create /pharmacy/register page. Form fields: name, address, phone, license number, location picker (map or address input). Validation. Submit button. Show "Pending approval" message after registration.

Task 9.3: Build admin verification dashboard API

- **Story Points:** 3
- **Labels:** `backend`, `admin`, `pharmacy`, `sprint-3`, `must-have`
- **Description:** Create GET /api/admin/pharmacies/pending endpoint (auth required, admin only). Return list of unverified pharmacies. Create PATCH /api/admin/pharmacies/{id}/verify endpoint to approve/reject.

Task 9.4: Create admin verification dashboard UI

- **Story Points:** 5
- **Labels:** `frontend`, `admin`, `pharmacy`, `sprint-3`, `must-have`, `ui`
- **Description:** Create /admin/verify-pharmacies page. Table showing: pharmacy name, address, phone, license, submitted date. Actions: Approve button, Reject button. Filter by pending/approved/rejected.

Task 9.5: Connect registration and verification flows

- **Story Points:** 3
- **Labels:** `frontend`, `backend`, `integration`, `pharmacy`, `admin`, `sprint-3`, `must-have`
- **Description:** Connect pharmacy form to registration API. Connect admin dashboard to verification API. Test: register pharmacy → admin sees it → admin approves → pharmacy appears on map.

STORY 10: Inventory Management Dashboard

Sprint: Sprint 3 (December 20 - December 26, 2024)
Epic: Inventory Management (PHARMY-6)
Story Points: 27
Labels: `backend`, `frontend`, `inventory`, `database`, `sprint-3`, `must-have`

User Story:
As a pharmacy owner
I want to manage my medicine inventory
So that users can see what medicines I have available

Acceptance Criteria:

- ✅ Pharmacy can view all their inventory
- ✅ Toggle availability on/off for each medicine
- ✅ Add notes to medicine entries
- ✅ Add new medicines to inventory
- ✅ Remove medicines from inventory
- ✅ See last updated timestamp
- ✅ View recent inventory changes history

Tasks:

Task 10.1: Create inventory management APIs (CRUD operations)

- **Story Points:** 8
- **Labels:** `backend`, `inventory`, `database`, `sprint-3`, `must-have`, `pair-programming`
- **Description:** Create: GET /api/pharmacy/inventory (list all), POST /api/pharmacy/inventory (add), PATCH /api/pharmacy/inventory/{id} (update), DELETE /api/pharmacy/inventory/{id} (delete). Include medicine details in responses.

Task 10.2: Implement automatic inventory_history logging

- **Story Points:** 3
- **Labels:** `backend`, `inventory`, `database`, `sprint-3`, `must-have`, `learning`
- **Description:** Create database trigger or application logic to log every inventory change to inventory_history table. Record: pharmacy_id, medicine_id, field_changed, old_value, new_value, timestamp.

Task 10.3: Create inventory dashboard UI with table

- **Story Points:** 5
- **Labels:** `frontend`, `inventory`, `sprint-3`, `must-have`, `ui`
- **Description:** Create /pharmacy/inventory page. Table columns: Medicine Name, Category, Available (toggle), Notes (editable), Last Updated, Actions (delete). Add "Add Medicine" button. Make table sortable and searchable.

Task 10.4: Implement toggle switches and editable notes

- **Story Points:** 3
- **Labels:** `frontend`, `inventory`, `ui`, `sprint-3`, `must-have`
- **Description:** Add toggle switch for availability (on/off). Make notes field inline-editable (click to edit, blur to save). Show loading state during save. Show success/error feedback.

Task 10.5: Add "Add medicine" functionality

- **Story Points:** 3
- **Labels:** `frontend`, `inventory`, `sprint-3`, `must-have`
- **Description:** Create modal/drawer for adding medicine. Dropdown to select from medicines table. Set initial availability and notes. Submit button. Refresh table after adding.

Task 10.6: Display recent inventory changes

- **Story Points:** 2
- **Labels:** `frontend`, `inventory`, `ui`, `sprint-3`, `should-have`
- **Description:** Show recent changes section below table. Display: medicine name, action (added/updated/deleted), timestamp. Show last 10 changes. Optional: add "View all history" link.

Task 10.7: Connect inventory dashboard to APIs

- **Story Points:** 3
- **Labels:** `frontend`, `backend`, `integration`, `inventory`, `sprint-3`, `must-have`
- **Description:** Connect all inventory operations to APIs. Test: add medicine, toggle availability, edit notes, delete medicine, view history. Verify all updates reflect on map search.

STORY 11: CSV Bulk Upload

Sprint: Sprint 3 (December 20 - December 26, 2024)
Epic: Inventory Management (PHARMY-6)
Story Points: 16
Labels: `backend`, `frontend`, `inventory`, `sprint-3`, `should-have`

User Story:
As a pharmacy owner
I want to upload my inventory via CSV
So that I can quickly add many medicines at once

Acceptance Criteria:

- ✅ Upload CSV file with medicine data
- ✅ System validates medicine names
- ✅ System creates/updates inventory records
- ✅ Changes logged to inventory_history
- ✅ Success/error report displayed

- ✅ Progress indicator during upload

Tasks:

Task 11.1: Build CSV parsing and upload API endpoint

- **Story Points:** 8
- **Labels:** `backend`, `inventory`, `sprint-3`, `should-have`, `pair-programming`, `learning`
- **Description:** Create POST /api/pharmacy/inventory/bulk-upload endpoint. Accept CSV file. Parse rows (medicine_name, available, notes). Validate medicine names exist in medicines table. Create/update inventory records. Log to history. Return summary (success count, error list).

Task 11.2: Create CSV upload UI with file picker

- **Story Points:** 3
- **Labels:** `frontend`, `inventory`, `sprint-3`, `should-have`, `ui`
- **Description:** Add "Bulk Upload" button to inventory page. Open modal with file input. Accept only .csv files. Show selected filename. Add "Upload" button. Provide CSV template download link.

Task 11.3: Add upload progress and success/error messages

- **Story Points:** 3
- **Labels:** `frontend`, `inventory`, `ui`, `sprint-3`, `should-have`
- **Description:** Show progress bar during upload. Display success message with count (e.g., "35 medicines added/updated"). Show detailed error list if any medicines failed. Allow dismissing errors.

Task 11.4: Test CSV upload with various file formats

- **Story Points:** 2
- **Labels:** `testing`, `inventory`, `sprint-3`, `should-have`
- **Description:** Test: valid CSV, CSV with invalid medicine names, CSV with missing columns, large CSV (100+ rows), CSV with special characters, empty CSV. Verify error handling.

SPRINT 4: REPORTS + POLISH & BUG FIXES + DEPLOYMENT

Duration: December 27, 2024 - January 1, 2025 (6 days)
Goal: Complete reports feature, fix all bugs, polish UI/UX, deploy, and prepare demo

STORY 12: User Reporting System

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Reports & Admin System (PHARMY-7)
Story Points: 10
Labels: `backend`, `frontend`, `reports`, `sprint-4`, `should-have`

User Story:
As a user
I want to report inaccurate pharmacy information
So that the data stays accurate and helpful

Acceptance Criteria:

- ✅ "Report inaccuracy" button on pharmacy popups
- ✅ Report form with type selection and description
- ✅ Reports submitted successfully
- ✅ User sees confirmation message

Tasks:

Task 12.1: Build reports submission API endpoint

- **Story Points:** 3
- **Labels:** `backend`, `reports`, `sprint-4`, `should-have`
- **Description:** Create POST /api/reports endpoint (auth required). Accept: pharmacy_id, report_type (wrong_availability, wrong_location, wrong_contact, other), description. Set status=pending. Return success.

Task 12.2: Create report form UI with type selection

- **Story Points:** 3
- **Labels:** `frontend`, `reports`, `ui`, `sprint-4`, `should-have`
- **Description:** Create report modal/drawer. Dropdown for report type. Textarea for description (required). Submit button. Show loading state. Display success message after submission.

Task 12.3: Add "Report inaccuracy" button to map popups

- **Story Points:** 2
- **Labels:** `frontend`, `reports`, `map`, `ui`, `sprint-4`, `should-have`
- **Description:** Add "Report Issue" button to pharmacy info popup. Open report modal on click. Pre-fill pharmacy_id. Make button small and non-intrusive.

Task 12.4: Connect report form to API

- **Story Points:** 2
- **Labels:** `frontend`, `backend`, `integration`, `reports`, `sprint-4`, `should-have`
- **Description:** Connect form submission to API. Handle success (close modal, show toast). Handle errors (display message). Clear form after successful submission.

STORY 13: Admin Reports Management

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Reports & Admin System (PHARMY-7)
Story Points: 13
Labels: `backend`, `frontend`, `admin`, `reports`, `sprint-4`, `should-have`

User Story:
As an admin
I want to review and resolve user reports
So that I can maintain data accuracy

Acceptance Criteria:

- ✅ Admin can view all reports (pending first)
- ✅ Admin can see report details
- ✅ Admin can add resolution notes
- ✅ Admin can resolve or dismiss reports
- ✅ Resolved reports marked appropriately

Tasks:

Task 13.1: Build admin reports APIs (list, resolve, dismiss)

- **Story Points:** 3
- **Labels:** `backend`, `admin`, `reports`, `sprint-4`, `should-have`
- **Description:** Create GET /api/admin/reports (list all, filter by status). Create PATCH /api/admin/reports/{id}/resolve (accept resolution_notes, set status=resolved). Create PATCH /api/admin/reports/{id}/dismiss.

Task 13.2: Create admin reports dashboard UI

- **Story Points:** 5
- **Labels:** `frontend`, `admin`, `reports`, `sprint-4`, `should-have`, `ui`
- **Description:** Create /admin/reports page. Table showing: pharmacy name, report type, description, submitted by, date, status. Filter tabs: All, Pending, Resolved,

Dismissed. Click row to view details.

Task 13.3: Add resolve/dismiss functionality

- **Story Points:** 3
- **Labels:** `frontend`, `admin`, `reports`, `sprint-4`, `should-have`
- **Description:** Create detail panel/modal when clicking report row. Show full details. Add textarea for resolution notes. Add "Resolve" and "Dismiss" buttons. Update list after action.

Task 13.4: Connect reports dashboard to APIs

- **Story Points:** 2
- **Labels:** `frontend`, `backend`, `integration`, `admin`, `reports`, `sprint-4`, `should-have`
- **Description:** Connect dashboard to APIs. Test: view reports, filter by status, resolve report with notes, dismiss report. Verify updates persist.

STORY 14: Comprehensive Testing & Bug Fixes

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Testing, Polish & Deployment (PHARMY-8)
Story Points: 32
Labels: `testing`, `qa`, `bug`, `sprint-4`, `must-have`

User Story:
As a team
I want to test all user flows and fix bugs
So that the application works reliably

Acceptance Criteria:

- ✅ All user flows tested end-to-end (patient, pharmacy, admin)
- ✅ Tested on multiple browsers (Chrome, Firefox, Safari)
- ✅ Tested on mobile devices
- ✅ All critical and high priority bugs fixed
- ✅ Performance tested with realistic data

Tasks:

Task 14.1: Test patient user flow end-to-end

- **Story Points:** 3

- **Labels:** `testing`, `qa`, `user`, `sprint-4`, `must-have`
- **Description:** Test complete patient flow: register → login → view map → search medicine → see results on map → view pharmacy popup → report inaccuracy → logout.

Task 14.2: Test pharmacy user flow end-to-end

- **Story Points:** 3
- **Labels:** `testing`, `qa`, `pharmacy`, `sprint-4`, `must-have`
- **Description:** Test complete pharmacy flow: register pharmacy → wait for admin approval → login → manage inventory (add, edit, delete, toggle) → CSV upload → view history.

Task 14.3: Test admin user flow end-to-end

- **Story Points:** 2
- **Labels:** `testing`, `qa`, `admin`, `sprint-4`, `must-have`
- **Description:** Test complete admin flow: login → verify pending pharmacies (approve/reject) → view reports → resolve/dismiss reports → verify changes reflect.

Task 14.4: Cross-browser testing (Chrome, Firefox, Safari)

- **Story Points:** 3
- **Labels:** `testing`, `qa`, `cross-browser`, `sprint-4`
- **Description:** Test all core features on Chrome, Firefox, Safari. Check layout, functionality, form submissions, map rendering, authentication.

Task 14.5: Mobile responsive testing

- **Story Points:** 3
- **Labels:** `testing`, `qa`, `mobile`, `responsive`, `sprint-4`
- **Description:** Test on iOS and Android devices. Test portrait and landscape. Verify map controls, search bar, forms, tables all work well on small screens.

Task 14.6: Performance testing with realistic data

- **Story Points:** 3
- **Labels:** `testing`, `performance`, `optimization`, `sprint-4`
- **Description:** Add 50+ pharmacies, 100+ medicines, 1000+ inventory records. Test search speed, map rendering performance, table load times. Identify bottlenecks.

Task 14.7: Create bug list and prioritize

- **Story Points:** 2
- **Labels:** `testing`, `qa`, `documentation`, `sprint-4`

- **Description:** Compile all bugs found during testing. Categorize as Critical (blocks core features), High (major issue), Medium (minor issue), Low (cosmetic). Create Jira tickets for each.

Task 14.8: Fix critical bugs

- **Story Points:** 8
- **Labels:** `bug`, `urgent`, `high-priority`, `sprint-4`, `must-have`
- **Description:** Fix all critical bugs that block core functionality or cause data loss/corruption. Test fixes thoroughly.

Task 14.9: Fix high priority bugs

- **Story Points:** 5
- **Labels:** `bug`, `high-priority`, `sprint-4`
- **Description:** Fix all high priority bugs that affect user experience or cause errors but don't block core features.

STORY 15: UI/UX Polish

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Testing, Polish & Deployment (PHARMY-8)
Story Points: 22
Labels: `frontend`, `design`, `ui`, `sprint-4`, `must-have`

User Story:
As a user
I want a polished and professional interface
So that the app is pleasant to use

Acceptance Criteria:

- ✅ Map interface matches prototype design
- ✅ Consistent styling across all pages
- ✅ Loading indicators on all async operations
- ✅ Success/error toast notifications
- ✅ User-friendly error messages
- ✅ Empty states for all lists
- ✅ Form validation feedback improved
- ✅ Smooth map interactions

Tasks:

Task 15.1: Ensure map matches prototype design

- **Story Points:** 3
- **Labels:** `frontend`, `design`, `ui`, `map`, `sprint-4`
- **Description:** Compare map interface to prototype. Adjust colors, marker styles, popup designs, search bar styling. Ensure exact match.

Task 15.2: Standardize spacing, typography, colors

- **Story Points:** 3
- **Labels:** `frontend`, `design`, `ui`, `sprint-4`
- **Description:** Create design tokens for spacing, typography, colors. Apply consistently across all pages. Use shadcn/ui theme variables.

Task 15.3: Add loading indicators to all async operations

- **Story Points:** 3
- **Labels:** `frontend`, `ui`, `sprint-4`
- **Description:** Add spinners/skeletons for: page loads, API calls, form submissions, table data loading, map data loading. Ensure no action appears "stuck".

Task 15.4: Implement toast notifications

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `sprint-4`
- **Description:** Add toast library (shadcn/ui toast component). Show for: successful actions (save, delete, submit), errors, warnings. Auto-dismiss after 3-5 seconds.

Task 15.5: Improve error messages

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `sprint-4`
- **Description:** Replace generic errors with user-friendly messages. Examples: "Email already exists" instead of "400 error", "Please check your internet connection" for network errors.

Task 15.6: Add empty states for lists/tables

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `sprint-4`
- **Description:** Add empty states with helpful icons and messages for: no inventory items, no reports, no search results, no pharmacies pending approval.

Task 15.7: Enhance form validation feedback

- **Story Points:** 2

- **Labels:** `frontend`, `ui`, `sprint-4`
- **Description:** Add real-time validation feedback. Show checkmarks for valid fields. Show error messages below fields. Highlight invalid fields in red. Disable submit until valid.

Task 15.8: Add tooltips and improve navigation

- **Story Points:** 2
- **Labels:** `frontend`, `ui`, `accessibility`, `sprint-4`
- **Description:** Add tooltips to buttons with icons. Add breadcrumbs to dashboard pages. Improve navigation menu. Add active state highlights.

Task 15.9: Optimize mobile map experience

- **Story Points:** 3
- **Labels:** `frontend`, `ui`, `mobile`, `responsive`, `optimization`, `sprint-4`
- **Description:** Make search bar collapsible on mobile. Adjust marker sizes for touch targets. Make sidebar swipeable. Ensure smooth pinch-to-zoom and panning.

STORY 16: Documentation & Sample Data

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Testing, Polish & Deployment (PHARMY-8)
Story Points: 13
Labels: `documentation`, `database`, `setup`, `sprint-4`, `must-have`

User Story:
As a team member
I want complete documentation
So that others can understand and deploy the project

Acceptance Criteria:

- ✅ README with setup instructions
- ✅ API endpoints documented with examples
- ✅ User guide created
- ✅ Admin guide created
- ✅ Deployment process documented
- ✅ 30 medicines in database
- ✅ 10+ realistic pharmacies across Beirut
- ✅ Sample reports created

Tasks:

Task 16.1: Write comprehensive README

- **Story Points:** 2
- **Labels:** `documentation`, `sprint-4`, `must-have`
- **Description:** Write README.md with: project description, tech stack, prerequisites, installation steps, environment variables, how to run dev servers, how to run tests.

Task 16.2: Document all API endpoints

- **Story Points:** 3
- **Labels:** `documentation`, `backend`, `sprint-4`, `must-have`
- **Description:** Create API.md documenting all endpoints: URL, HTTP method, auth required, request body example, response example, error codes. Use tools like Postman or Swagger if available.

Task 16.3: Create user and admin guides

- **Story Points:** 3
- **Labels:** `documentation`, `sprint-4`
- **Description:** Write USER_GUIDE.md (how to search medicines, report issues) and ADMIN_GUIDE.md (how to verify pharmacies, manage reports). Include screenshots.

Task 16.4: Document deployment process

- **Story Points:** 2
- **Labels:** `documentation`, `deployment`, `devops`, `sprint-4`
- **Description:** Write DEPLOYMENT.md with step-by-step instructions: setting up Netlify, Railway/Render, configuring production database, environment variables, build commands, post-deployment checks.

Task 16.5: Add realistic sample data (medicines, pharmacies)

- **Story Points:** 3
- **Labels:** `database`, `setup`, `sprint-4`
- **Description:** Create seeds for 30 common medicines (Panadol, Aspirin, etc.) with categories. Create 10-15 realistic pharmacies across Beirut with real-looking addresses, coordinates, phone numbers.

STORY 17: Production Deployment

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Testing, Polish & Deployment (PHARMY-8)

Story Points: 23
Labels: `deployment`, `devops`, `frontend`, `backend`, `database`, `must-have`

User Story:
As a team
I want to deploy the application to production
So that it's publicly accessible

Acceptance Criteria:

- ✅ Frontend deployed to Netlify
- ✅ Backend deployed to Railway/Render
- ✅ Production database setup
- ✅ Environment variables configured
- ✅ Frontend connected to production API
- ✅ All features work in production
- ✅ No deployment-specific issues

Tasks:

Task 17.1: Setup hosting platforms (Netlify, Railway/Render)

- **Story Points:** 2
- **Labels:** `deployment`, `devops`, `setup`, `must-have`
- **Description:** Create accounts on Netlify and Railway/Render. Connect GitHub repository. Configure build settings for Next.js and Laravel.

Task 17.2: Create and configure production database

- **Story Points:** 3
- **Labels:** `deployment`, `database`, `devops`, `must-have`
- **Description:** Create PostgreSQL database on Railway/Render or separate service (e.g., Supabase, ElephantSQL). Note connection string. Run migrations. Add sample data.

Task 17.3: Deploy backend API to production

- **Story Points:** 5
- **Labels:** `deployment`, `backend`, `devops`, `must-have`
- **Description:** Configure Laravel for production (APP_ENV=production, APP_DEBUG=false). Set up database connection. Configure CORS for frontend domain. Deploy to Railway/Render. Test API endpoints.

Task 17.4: Deploy frontend to Netlify

- **Story Points:** 3
- **Labels:** `deployment`, `frontend`, `devops`, `must-have`
- **Description:** Configure Next.js build settings. Set API base URL to production backend. Deploy to Netlify. Set up custom domain if available. Enable HTTPS.

Task 17.5: Configure environment variables

- **Story Points:** 2
- **Labels:** `deployment`, `devops`, `setup`, `security`, `must-have`
- **Description:** Set all environment variables on hosting platforms: DB credentials, JWT secret, API keys (maps), CORS origins, APP_KEY. Verify all are set correctly.

Task 17.6: Test deployed application thoroughly

- **Story Points:** 5
- **Labels:** `deployment`, `testing`, `qa`, `must-have`
- **Description:** Test complete user flows on production: registration, login, map display, search, inventory management, admin features. Test on multiple devices and browsers.

Task 17.7: Fix deployment-specific issues

- **Story Points:** 3
- **Labels:** `deployment`, `bug`, `devops`
- **Description:** Fix any issues that only appear in production: CORS errors, asset loading issues, API connection problems, routing issues, database connection errors.

STORY 18: Demo Preparation

Sprint: Sprint 4 (December 27, 2024 - January 1, 2025)
Epic: Testing, Polish & Deployment (PHARMY-8)
Story Points: 8
Labels: `documentation`, `testing`, `sprint-4`, `must-have`

User Story:
As a team
I want to prepare for the demo showcase
So that we can effectively present our project during the Google Meet

Acceptance Criteria:

- ✅ Demo flow script prepared

- ✅ All demo scenarios tested and working
- ✅ Production environment ready with sample data
- ✅ Team practiced demo at least once

Tasks:

Task 18.1: Prepare demo flow script

- **Story Points:** 2
- **Labels:** `documentation`, `sprint-4`, `must-have`
- **Description:** Write step-by-step demo script covering: user registration/login, medicine search on map, pharmacy inventory management, admin approval workflow. Keep it concise for Google Meet showcase.

Task 18.2: Test all demo scenarios

- **Story Points:** 3
- **Labels:** `testing`, `sprint-4`, `must-have`
- **Description:** Run through complete demo flow: patient searching for medicine, pharmacy managing inventory, admin approving pharmacies. Ensure everything works smoothly end-to-end.

Task 18.3: Prepare production environment for demo

- **Story Points:** 2
- **Labels:** `deployment`, `database`, `sprint-4`, `must-have`
- **Description:** Clear test/dummy data from production. Add realistic sample data (medicines, pharmacies, inventory). Ensure production is stable and ready for showcase.

Task 18.4: Practice demo walkthrough

- **Story Points:** 1
- **Labels:** `sprint-4`, `must-have`
- **Description:** Practice demo flow at least once as a team. Assign roles for who demonstrates what. Ensure smooth transitions. Test screen sharing on Google Meet.

JIRA CONFIGURATION

Workflow States

1. **To Do** - Task created, not started

2. **In Progress** - Actively being worked on
3. **Code Review** - Waiting for peer review
4. **Testing** - Being tested
5. **Done** - Completed and verified

Components to Create

- Frontend (Next.js)
- Backend (Laravel)
- Database (PostgreSQL)
- DevOps & Deployment
- Documentation

Custom Fields (Optional)

- **Priority:** Must Have / Should Have / Could Have
- **Sprint Goal:** Brief description
- **Blocked By:** Track dependencies

Estimation Guidelines

- **1-2 points:** Small UI fix, minor documentation
- **3-5 points:** Medium feature, standard CRUD API
- **8 points:** Complex integration, major feature
- **13 points:** Very complex system, multiple integrations

DAILY WORKFLOW

Every Morning:

- 15-minute standup via Google Meet
- Each member shares: what I did yesterday, what I'm doing today, any blockers

Throughout the Day:

- Update task status in Jira as you work
- Move tasks: To Do → In Progress → Code Review → Testing → Done
- Add comments to tasks with progress updates

When Blocked:

- Mark task with `blocked` label

- Post immediately in Discord/Slack `#blockers` channel
- Tag relevant team member who can help

Before Merging Code:

- Get at least one code review approval
- Ensure all tests pass
- Update Jira task to "Done"

End of Sprint:

- **Sprint Review:** Demo completed features
- **Sprint Retrospective:** What went well, what to improve

CRITICAL SUCCESS FACTORS

Must Have Features (Non-negotiable)

✅ User registration/login (all roles)
✅ Map-based interface matching prototype
✅ Medicine search with map integration
✅ Pharmacy display on map with availability
✅ Pharmacy registration with admin verification
✅ Basic inventory management
✅ Admin dashboard

Should Have (Cut if behind schedule)

⚠️ CSV bulk upload
⚠️ Reports system
⚠️ Inventory history tracking

Could Have (Nice to have)

💡 Email notifications
💡 Password reset
💡 Advanced filters

DATABASE TABLES REFERENCE

1. **users** - All user accounts (patients, pharmacies, admins)

2. **pharmacies** - Pharmacy business information
3. **medicines** - Master list of medicines
4. **inventory** - Pharmacy medicine inventory (what each pharmacy has)
5. **reports** - User-submitted reports about pharmacy inaccuracies
6. **inventory_history** - Audit log of inventory changes

TECHNOLOGY STACK

Frontend:

- Next.js 14+ with TypeScript
- shadcn/ui component library
- Tailwind CSS
- Google Maps / Mapbox
- React Context for auth

Backend:

- Laravel (PHP)
- JWT authentication
- RESTful API

Database:

- PostgreSQL

Deployment:

- Frontend: Netlify
- Backend: Railway / Render
- Database: Railway / Render / Supabase

Tools:

- Git & GitHub
- Jira (project management)
- Discord/Slack (communication)
- Google Meet (standups)
- Figma (design)

QUICK LABEL REFERENCE

Mandatory Labels for Every Task:

1. At least ONE technology label: `frontend`, `backend`, `database`, `integration`, or `devops`
2. At least ONE sprint label: `sprint-0`, `sprint-1`, `sprint-2`, `sprint-3`, `sprint-4`, `deployment`, or `presentation`
3. At least ONE priority label: `must-have`, `should-have`, or `could-have`

Optional but Recommended:

- Feature domain: `authentication`, `map`, `search`, `inventory`, `reports`, `admin`, `pharmacy`
- Work type: `setup`, `design`, `documentation`, `testing`, `bug`, `refactor`, `optimization`
- Special: `urgent`, `blocked`, `pair-programming`, `ai-assisted`, `learning`, `security`, `performance`

LABEL FILTERING TIPS

See all authentication work:
`label = authentication`

See Sprint 2 incomplete tasks:
`label = sprint-2 AND status != Done`

See critical/urgent items:
`label IN (urgent, must-have)`

See backend bugs:
`label = bug AND label = backend`

See blocked items:
`label = blocked`

See available frontend tasks:
`label = frontend AND status = "To Do"`

PROJECT TIMELINE SUMMARY

Sprint	Duration	Days	Focus
Sprint 0	Nov 29 - Dec 5	7	Foundation & Setup
Sprint 1	Dec 6 - Dec 12	7	Authentication + Map
Sprint 2	Dec 13 - Dec 19	7	Medicine Search
Sprint 3	Dec 20 - Dec 26	7	Pharmacy + Inventory
Sprint 4	Dec 27 - Jan 1	6	Reports + Polish + Deploy + Demo Prep

Total: 34 days

NOTES FOR JIRA SETUP

1. **Create Epics First** - All 8 epics (PHARMY-1 through PHARMY-8)
2. **Create All Labels** - Use the labels library above
3. **Create Sprints** - Sprint 0 through Sprint 4
4. **Add Stories to Epics** - 18 stories total across 8 epics
5. **Add Tasks to Stories** - Each task linked to parent story
6. **Assign Labels** - Use the label assignments provided for each task
7. **Set Story Points** - Use the points provided for each story/task
8. **Configure Workflow** - Set up the 5 workflow states
9. **Start Sprint 0** - Activate Sprint 0 and begin work!

Last Updated: November 29, 2024
Document Version: 1.1
Prepared for: Pharmy Project Team
Project Deadline: January 1, 2025

END OF DOCUMENT

This complete workflow guide contains all information needed to set up and manage the Pharmy project in Jira. Keep this document handy throughout the project for reference on sprint goals, task details, labels, and workflow guidelines.

IMPORTANT NOTE: The project deadline is January 1, 2025. There will be no formal presentation - the final deliverable is a working deployed application showcased during a Google Meet session.