Board Module – End-to-End Acceptance & Integration Testing Document

1. Sanity Testing (Multi-User Validation)

- Given two users (User A and User B) are registered and logged in separately, when each creates a project, then their projects must appear independently in their own dashboard.
- Given User A creates a task in Project X, when User B logs in, then User B should not see that task.
- Given both users perform CRUD operations, when refreshing the UI, then only their own data should persist from the backend.
- API validation: Verify that requests include a valid user_id token and return status 200/201/204 depending on operation.

2. Project Management

- Given a user clicks 'Add Project' and enters a name, when Save is clicked, then the new project appears in the sidebar.
- Given a project exists, when it's deleted, then it disappears from both UI and backend (verify DELETE /api/v1/projects/{id}).
- Given multiple projects exist, when navigating between them, then the board view updates accordingly without stale data.
- Network Tab Check: Ensure payload { name: <string> } and response 201 Created with project_id.

3. Board Columns

- Given I open a project board, when I view columns, then I should see To Do, In Progress, Validation, and Done.
- Given I drag a task between columns, when dropping, then the status must update via PATCH /api/v1/tasks/{id}/status.
- Given I rename or delete a column, when saving, then order should persist in local state and backend on reload.
- Expected response: 200 OK, column order consistent after refresh.

4. Task / User Story Management

- Given I click 'Add Task', when I fill all required fields and select a Feature, then the task is created and linked properly.
- Given I edit an existing story, when I save, then fields (title, description, assignee, status) are updated (PATCH /api/v1/tasks/{id}).

- Given I delete a story, when confirming, then it disappears from UI and database (DELETE /api/v1/tasks/{id}).
- Given I move a task to 'Done', when refreshing, then it remains under Done column (persisted).

5. Feature Dashboard

- Given I open Feature Dashboard, when I click 'New Feature', then a modal opens to enter details.
- Given I fill Feature Name, Details, User Story, Core Requirements, Acceptance Criteria, and Technical Notes, then Save creates a new record (POST /api/v1/features).
- Given I expand a Feature row, then I should see all linked stories displayed with story_code, title, and status badge.
- Given I delete a feature, when confirming, then it is removed and related tasks unlinked.

6. Feature Form

- Given I open Feature Form modal, when I fill all fields and click Save, then new feature persists (POST /api/v1/features).
- Given I leave any required field blank, when clicking Save, then validation messages appear, and no API call fires.
- Given I click Cancel, when modal closes, then no change should be reflected in backend.
- Check Network tab for request payload with all feature fields and verify 201 Created.

7. Integration & Data Persistence

- Given I create or edit entities, when refreshing, then UI must reload data from backend APIs correctly.
- Given I inspect network tab, then all create/update/delete actions return expected codes (200, 201, 204).
- Given multiple users perform updates, when tested simultaneously, then data isolation is maintained (verify different user_id payloads).

8. UI/UX & Accessibility

- Given I open any modal, then focus is trapped inside until closed (keyboard accessible).
- Given I resize browser, then modals and grids respond correctly for mobile/tablet/desktop.
- Given I navigate using Tab key, then all actionable items are reachable and visible focus states exist.

9. Export / Reporting

- Given I click 'Export Stories' button, when API succeeds, then a file should download containing correct data.

- Given exported CSV/XLS contains fields Title, Status, Feature, Assignee, Priority, CreatedAt, then all data must match UI values.

10. Regression and Smoke Testing

- After every new deployment, verify board loads correctly and all modals are accessible.
- Create → Edit → Delete flow works without refresh requirement.
- Project switching should not retain tasks or features from previous context.
- Check console for no JavaScript errors or API failures (Network tab: 2xx responses).

Appendix: API Verification

- POST /api/v1/projects
- GET /api/v1/projects
- POST /api/v1/tasks
- PATCH /api/v1/tasks/{task_id}/status
- PATCH /api/v1/tasks/{task_id}
- DELETE /api/v1/tasks/{task_id}
- GET /api/v1/tasks/feature/{feature_id}
- POST /api/v1/features
- PATCH /api/v1/features/{feature_id}
- DELETE /api/v1/features/{feature_id}
- GET /api/v1/features/project/{project_id}