DEPARTMENT OF SOFTWARE ENGINEERING
Faculty of Computing & Information Technology (FCIT)
University of the Punjab, Lahore – PAKISTAN

FYDP-DSE D1

Agile: Product Backlog & Definition of Done

Smart Shopping Cart

Bachelors of Science in Software Engineering (2022-2026)

By

Laiba Khalid BSEF22M504

Haiqa Khan BSEF22M507

Ayesha Shahid BSEF22M528

Supervised by:

Primary supervisor's name,
Dr Madiha Khalid

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, UNIVERSITY OF THE PUNJAB, LAHORE.

Contents

1. Product Vision

A smart shopping cart system for retail customers that eliminates long checkout queues, prevents overspending, and improves inventory handling. By combining real-time purchase tracking with AI-powered budget-friendly recommendations, it helps shoppers save money and time while enabling retailers to automate stock management and improve operational efficiency.

2. Defined Epics & Features

Epic 1: Design Scan Interface (UI Only)

- **Feature:** Build a responsive scan interface layout with a scan button, product area, and alert section.
- **Feature:** Develop a reusable React component (ScannerInterface.jsx) with simulated scan functionality using sample products.
- **Feature:** Display and manage cart items with alerts and confirmation pop-ups for item removal and cart clearing.
- **Feature:** Ensure clean, consistent, and responsive code with testing and updated README documentation.

Epic 2: Customer Interface

- **Feature**: View live catalogue of items
- **Feature**: Scan items using barcode/RFID
- **Feature**: Display product details (name, price, stock availability)
- **Feature:** Show real-time bill total and item count

Epic 3: Total Calculation Module

- **Feature:** Automatic total price calculation
- **Feature:** Update totals when items are added/removed
- **Feature**: Budget limit indicator

Epic 4: Inventory Management

- **Feature:** Real-time stock updates with central database
- **Feature:** Low-stock threshold detection
- **Feature:** Inventory staff dashboard for restocking

Epic 5: Notification & Alerts Module

- **Feature:** Automated alerts for inventory managers
- **Feature:** Supplier notifications for critical shortages
- **Feature:** Dashboard alerts for managers

Epic 6: AI Recommendation Module

- **Feature:** Budget-friendly product suggestions
- **Feature:** Personalized recommendations based on cart items*/

3. Initial Product Backlog (Stories with Priority)

Priority	User Story
Must	As a customer, I want a scan interface UI so that I can view scanned products on screen.
Must	As a developer, I want to create a React component for the scan interface (UI only) so that it's modular, reusable, and easy to maintain.
Must	As a tester, I want to verify responsiveness and element alignment so that the interface looks correct and consistent on all screen sizes.
Must	As a team member, I want to commit the code, update the README, and mark the Jira tasks as done so that the work is properly documented and traceable.
Should	As a user, the scan interface should clearly show alert or message notifications (such as successful scans, item removals, or cart clearing).
Should	As a developer, the scan interface should follow proper naming conventions and clean code structure to remain maintainable.
Could	Implement a simulated scanning function to mimic adding items via barcode/RFID input (UI-level prototype).
Won't (for now)	Fetch product details from the database or API for each scanned item (to be implemented later during hardware integration).

4. Definition of Done (DoD) Checklist

DoD is the team's agreement on what "done" means. All stories/features must meet this before being considered complete.

Design Scan Interface

- React component for the scan interface is created (UI only, no hardware logic yet).
- Layout includes a scan button/icon, product display area, and message/alert section.
- Code follows clean structure and naming conventions.
- UI tested in browser for responsiveness and correct element alignment.
- Code pushed and committed to the Git repository.
- Peer/team review completed (if applicable).
- README updated with component purpose and usage instructions.
- Jira task marked as "Done."

5. Git Repo & Branching Strategy

Our team **will use GitHub** for version control, collaborative development, and continuous integration/deployment (CI/CD) automation through **GitHub Actions**.
The following branching model will ensure structured development, high code quality, and efficient deployment management.

Main Branch (main)

- It will contain only production-ready code.
- It will act as the protected branch, where direct commits will be prohibited.
- All updates will be made through Pull Requests (PRs) that will need to pass review and CI checks before merging.

Develop Branch (develop)

- It will maintain the latest integrated version of all completed and tested features.
- It will serve as a staging environment before code is merged into the main branch.
- Every merge will trigger automated build validations and unit tests through GitHub Actions.

Feature Branches (feature/<feature-name>)

- These branches will be used to develop individual features independently.
- They will always be created from the develop branch.
- After development and testing, each feature branch will be merged back into develop through a reviewed Pull Request (PR).

Hotfix Branches (hotfix/<issue-name>)

- These branches will be used for urgent production fixes.
- They will be created directly from the main branch.
- Once the fix is completed, the branch will be merged into both main and develop to maintain synchronization.

Branch Protection Rules

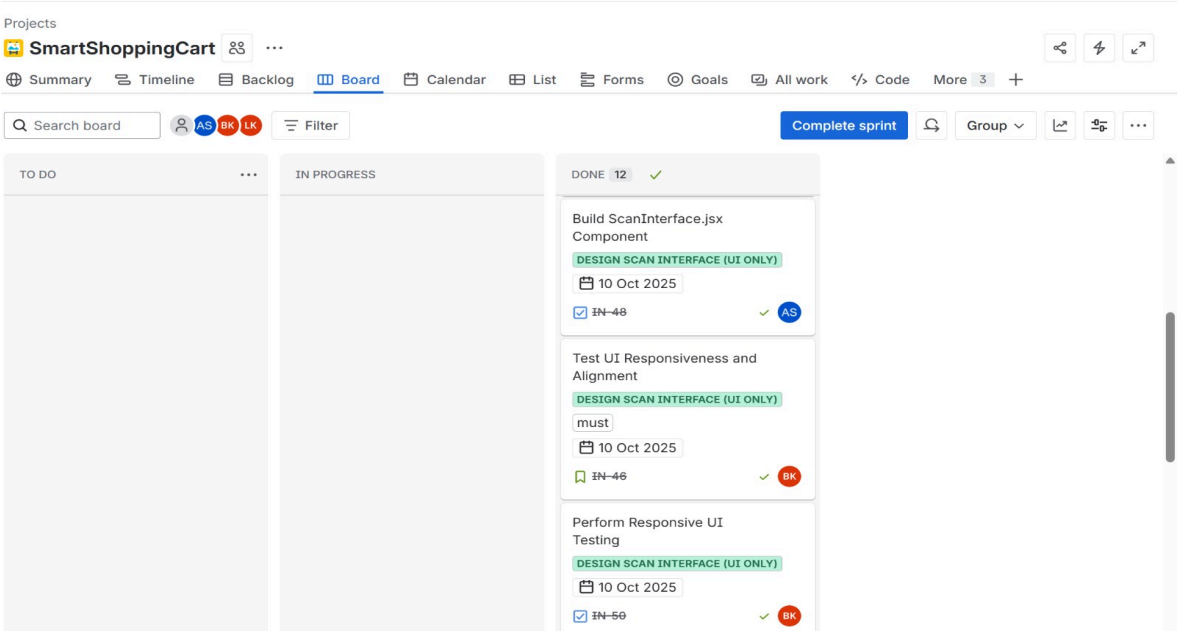
- Direct commits to main and develop will not be allowed.
- All merges will occur through Pull Requests (PRs).
- Each PR will require at least one team member's review and approval.
- GitHub Actions will automatically run tests, lint checks, and builds before a merge is approved.
- Merge conflicts will need to be resolved locally before final submission.

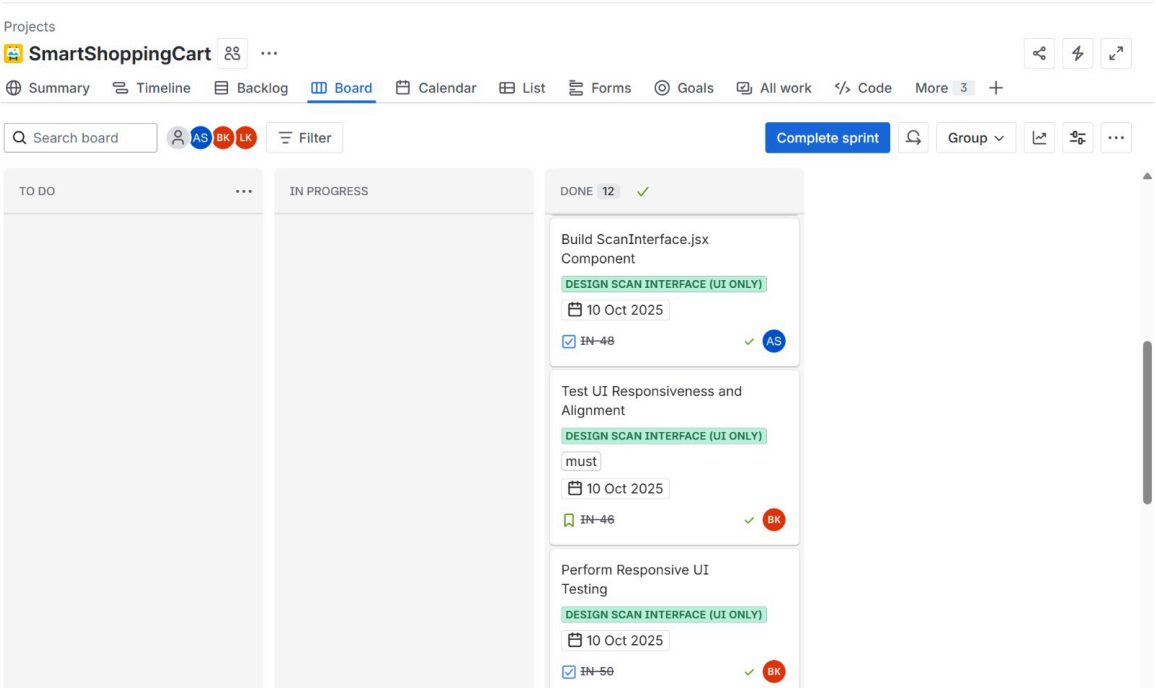
6. Jira Board Screenshot (if used)

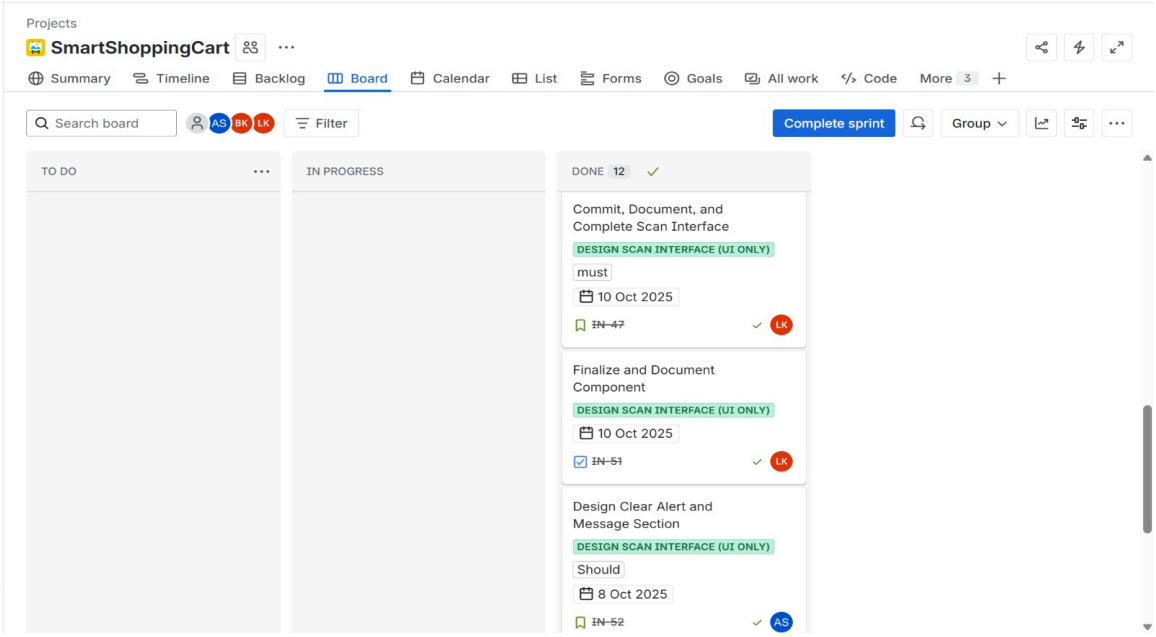
Below are the screenshots of our Jira Scrum Board for Epic: Design Scan Interface (UI Only).All user stories and tasks for this sprint are shown under the Done column, indicating completion of work as per the Definition of Done (DOD)

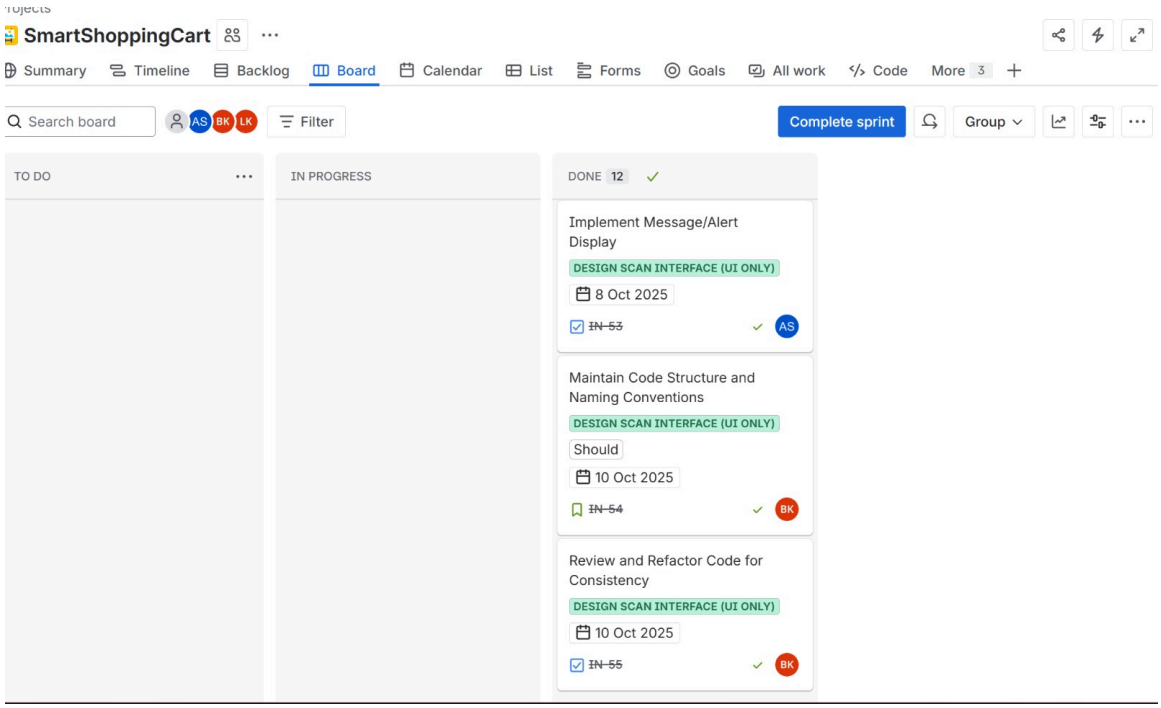
Jira Board Link:

https://pucit-team-wivu0h4p.atlassian.net/jira/software/projects/IN/boards/5




Projects
SmartShoppingCart
Summary
Timeline
Backlog
Board
Calendar
List
Forms
Goals
All work
Code
More 3
Search board
Filter
Complete sprint
Group
TO DO
IN PROGRESS
DONE 12
Build ScanInterface.jsx Component
DESIGN SCAN INTERFACE (UI ONLY)
10 Oct 2025
IN-48
Test UI Responsiveness and Alignment
DESIGN SCAN INTERFACE (UI ONLY)
must
10 Oct 2025
IN-46
Perform Responsive UI Testing
DESIGN SCAN INTERFACE (UI ONLY)
10 Oct 2025
IN-50


Projects
SmartShoppingCart
Summary
Timeline
Backlog
Board
Calendar
List
Forms
Goals
All work
Code
More 3
Search board
Filter
Complete sprint
Group
TO DO
IN PROGRESS
DONE 12
Commit, Document, and Complete Scan Interface
DESIGN SCAN INTERFACE (UI ONLY)
must
10 Oct 2025
IN-47
Finalize and Document Component
DESIGN SCAN INTERFACE (UI ONLY)
10 Oct 2025
IN-51
Design Clear Alert and Message Section
DESIGN SCAN INTERFACE (UI ONLY)
Should
8 Oct 2025
IN-52



7. Planned GitHub Action Flow Pipeline

Links

- The repository link is (https://github.com/haiqakhan1/smartshoppingcart)

Workflow Triggers

The pipeline will run automatically on:

- Every push to develop or main.
- Every pull request targeting develop or main.

Automated Steps

1. Checkout the latest code from the repository.
2. Install dependencies:
    - npm ci → for frontend (React).
    - dotnet restore → for backend (ASP.NET Core).
3. Run ESLint for frontend code quality checks.
4. Execute automated tests:
    - Frontend: npm test (Jest).
    - Backend: dotnet test (xUnit).
5. Build both applications:
    - npm run build → frontend.
    - dotnet build and dotnet publish → backend.

6. Run security and vulnerability scans on all dependencies.
7. Deploy automatically to staging (on the develop branch) after successful checks.
8. Deploy to production (on the main branch) after all checks pass and deployment approval is granted.