```
// script.js

// --- Global Data & Initial Setup ---
let products = []; // Stores product master data and current stock
let stockMovements = []; // Stores all historical movements

// DOM Elements
const stockSummaryBody = document.getElementById('stockSummaryBody');
const stockCardBody = document.getElementById('stockCardBody');
const stockCardProductSelect = document.getElementById('stockCardProductSelect');

const receiveModal = document.getElementById('receiveModal');
const dispatchModal = document.getElementById('dispatchModal');

const openReceiveModalBtn = document.getElementById('openReceiveModal');
const closeReceiveModalBtn = document.getElementById('closeReceiveModal');
const receiveForm = document.getElementById('receiveForm');
const receiveProductIdSelect = document.getElementById('receiveProductId');
const receiveQuantityInput = document.getElementById('receiveQuantity');
const receiveNoteInput = document.getElementById('receiveNote');

const openDispatchModalBtn = document.getElementById('openDispatchModal');
const closeDispatchModalBtn = document.getElementById('closeDispatchModal');
const dispatchForm = document.getElementById('dispatchForm');
const dispatchProductIdSelect = document.getElementById('dispatchProductId');
const dispatchQuantityInput = document.getElementById('dispatchQuantity');
const dispatchNoteInput = document.getElementById('dispatchNote');

const btnDashboard = document.getElementById('btnDashboard');
const btnStockCard = document.getElementById('btnStockCard');
const dashboardSection = document.getElementById('dashboard');
const stockCardSection = document.getElementById('stockCard');

// --- Utility Functions ---

// Load data from Local Storage
function loadData() {
    const storedProducts = localStorage.getItem('products');
    const storedMovements = localStorage.getItem('stockMovements');

    products = storedProducts ? JSON.parse(storedProducts) : [
        // Initial dummy data if no products found
        { id: 'SNG001', name: 'เบียร์สิงห์ กระป๋อง 320ml', unit: 'ลัง', currentStock: 100 },
        { id: 'SNG002', name: 'น้ำดื่มสิงห์ ขวดใหญ่ 1.5L', unit: 'แพ็ค', currentStock: 250 },
        { id: 'SNG003', name: 'โซดาสิงห์ ขวดเล็ก', unit: 'ลัง', currentStock: 80 }
    ];
    stockMovements = storedMovements ? JSON.parse(storedMovements) : [];
}
```

```
// Save data to Local Storage
function saveData() {
   localStorage.setItem('products', JSON.stringify(products));
   localStorage.setItem('stockMovements', JSON.stringify(stockMovements));
}

// Generate unique ID
function generateId(prefix) {
   return prefix + Date.now().toString(36) + Math.random().toString(36).substr(2,
5).toUpperCase();
}

// Format date and time
function formatDateTime(dateString) {
   const date = new Date(dateString);
   const options = { year: 'numeric', month: '2-digit', day: '2-digit', hour: '2-digit', minute:
'2-digit', second: '2-digit' };
   return date.toLocaleDateString('th-TH', options);
}

// Show a toast notification
function showToast(message, type = 'info') {
   const toastContainer = document.querySelector('.toast-container') || (() => {
      const div = document.createElement('div');
      div.className = 'toast-container';
      document.body.appendChild(div);
      return div;
   })();

   const toast = document.createElement('div');
   toast.className = `toast ${type}`;
   toast.textContent = message;
   toastContainer.appendChild(toast);

   setTimeout(() => {
      toast.remove();
   }, 4000); // Remove after 4 seconds (includes animation time)
}

// --- Render Functions ---

// Render stock summary table on Dashboard
function renderStockSummary() {
   stockSummaryBody.innerHTML = ''; // Clear existing rows
   products.forEach(product => {
      const row = stockSummaryBody.insertRow();
      row.innerHTML = `
```

```
            <td>${product.id}</td>
            <td>${product.name}</td>
            <td>${product.unit}</td>
            <td>${product.currentStock}</td>
            <td>
                <button class="small-action-btn view-stock-card" data-product-id="${product.id}"
title="ดูประวัติ"><i class="fas fa-eye"></i></button>
            </td>
        `;
    });

    // Add event listeners for "View Stock Card" buttons
    document.querySelectorAll('.view-stock-card').forEach(button => {
        button.addEventListener('click', (event) => {
            const productId = event.currentTarget.dataset.productId;
            switchSection('stockCard');
            stockCardProductSelect.value = productId;
            renderStockCard();
        });
    });
}

// Populate product dropdowns for modals and stock card filter
function populateProductDropdowns() {
    const productOptionsHtml = products.map(p => `<option value="${p.id}">${p.name}
(${p.id})</option>`).join('');
    receiveProductIdSelect.innerHTML = productOptionsHtml;
    dispatchProductIdSelect.innerHTML = productOptionsHtml;
    stockCardProductSelect.innerHTML = `<option value="">--- เลือกสินค้า ---</option>` +
productOptionsHtml;
}

// Render stock card for a selected product
function renderStockCard() {
    const selectedProductId = stockCardProductSelect.value;
    stockCardBody.innerHTML = ''; // Clear existing rows

    if (!selectedProductId) {
        return; // No product selected
    }

    const filteredMovements = stockMovements.filter(m => m.productId ===
selectedProductId)
                                        .sort((a, b) => new Date(a.date) - new Date(b.date)); // Sort by
date

    if (filteredMovements.length === 0) {
        stockCardBody.innerHTML = `<tr><td colspan="7" style="text-align:center;">ไม่พบ
ประวัติการเคลื่อนไหวสำหรับสินค้านี้</td></tr>`;
        return;
    }

    filteredMovements.forEach(movement => {
        const row = stockCardBody.insertRow();
        const typeClass = movement.type === 'รับเข้า' ? 'text-success' : 'text-danger';
        row.innerHTML = `
            <td>${formatDateTime(movement.date)}</td>
            <td class="${typeClass}">${movement.type}</td>
            <td>${movement.quantity}</td>
            <td>${movement.unit}</td>
            <td>${movement.stockBefore}</td>
            <td>${movement.stockAfter}</td>
            <td>${movement.note || '-'}</td>
        `;
    });
}

// --- Modal & Form Handlers ---

function openModal(modal) {
    modal.classList.add('active');
    // Set first product as default for dropdowns if available
    if (products.length > 0) {
        if (modal === receiveModal) {
            receiveProductIdSelect.value = products[0].id;
            receiveQuantityInput.value = '';
            receiveNoteInput.value = '';
        } else if (modal === dispatchModal) {
            dispatchProductIdSelect.value = products[0].id;
            dispatchQuantityInput.value = '';
            dispatchNoteInput.value = '';
        }
    }
}

function closeModal(modal) {
    modal.classList.remove('active');
}

// Handle Receive Form Submission
receiveForm.addEventListener('submit', (event) => {
    event.preventDefault(); // Prevent default form submission

    const productId = receiveProductIdSelect.value;
    const quantity = parseInt(receiveQuantityInput.value);
    const note = receiveNoteInput.value.trim();

    if (!productId || isNaN(quantity) || quantity <= 0) {
        showToast('กรุณากรอกข้อมูลให้ถูกต้อง', 'error');
        return;
    }

    const product = products.find(p => p.id === productId);
    if (product) {
        const stockBefore = product.currentStock;
        product.currentStock += quantity;

        const newMovement = {
            movementId: generateId('MOV'),
            productId: productId,
            type: 'รับเข้า',
            date: new Date().toISOString(),
            quantity: quantity,
            unit: product.unit,
            stockBefore: stockBefore,
            stockAfter: product.currentStock,
            note: note
        };
        stockMovements.push(newMovement);

        saveData();
        renderStockSummary();
        populateProductDropdowns(); // Update dropdowns if new products were added
(though not implemented here for simplicity)
        closeModal(receiveModal);
        showToast('บันทึกรับเข้าสินค้าสำเร็จ!', 'success');
    } else {
        showToast('ไม่พบสินค้าที่ระบุ', 'error');
    }
});

// Handle Dispatch Form Submission
dispatchForm.addEventListener('submit', (event) => {
    event.preventDefault(); // Prevent default form submission

    const productId = dispatchProductIdSelect.value;
    const quantity = parseInt(dispatchQuantityInput.value);
    const note = dispatchNoteInput.value.trim();

    if (!productId || isNaN(quantity) || quantity <= 0) {
        showToast('กรุณากรอกข้อมูลให้ถูกต้อง', 'error');
        return;
    }
    const product = products.find(p => p.id === productId);
    if (product) {
        if (product.currentStock < quantity) {
            showToast(`สต็อกไม่พอ! มีอยู่ ${product.currentStock} ${product.unit}`, 'error');
            return;
        }

        const stockBefore = product.currentStock;
        product.currentStock -= quantity;

        const newMovement = {
            movementId: generateId('MOV'),
            productId: productId,
            type: 'จ่ายออก',
            date: new Date().toISOString(),
            quantity: quantity,
            unit: product.unit,
            stockBefore: stockBefore,
            stockAfter: product.currentStock,
            note: note
        };
        stockMovements.push(newMovement);

        saveData();
        renderStockSummary();
        closeModal(dispatchModal);
        showToast('บันทึกจ่ายออกสินค้าสำเร็จ!', 'success');
    } else {
        showToast('ไม่พบสินค้าที่ระบุ', 'error');
    }
});

// --- Event Listeners ---

// Modal open/close events
openReceiveModalBtn.addEventListener('click', () => openModal(receiveModal));
closeReceiveModalBtn.addEventListener('click', () => closeModal(receiveModal));
openDispatchModalBtn.addEventListener('click', () => openModal(dispatchModal));
closeDispatchModalBtn.addEventListener('click', () => closeModal(dispatchModal));

// Close modal when clicking outside of content
window.addEventListener('click', (event) => {
    if (event.target === receiveModal) {
        closeModal(receiveModal);
    }
    if (event.target === dispatchModal) {
        closeModal(dispatchModal);
```

```
  }
});

// Navigation between sections
btnDashboard.addEventListener('click', () => switchSection('dashboard'));
btnStockCard.addEventListener('click', () => switchSection('stockCard'));

function switchSection(sectionId) {
  document.querySelectorAll('.app-section').forEach(section => {
    section.classList.remove('active');
  });
  document.getElementById(sectionId).classList.add('active');

  document.querySelectorAll('.nav-button').forEach(button => {
    button.classList.remove('active');
  });
  if (sectionId === 'dashboard') {
    btnDashboard.classList.add('active');
    renderStockSummary(); // Re-render to ensure latest data
  } else if (sectionId === 'stockCard') {
    btnStockCard.classList.add('active');
    populateProductDropdowns(); // Ensure dropdown is up-to-date
    renderStockCard(); // Render initially or if selected product changes
  }
}

// Event listener for product selection in Stock Card
stockCardProductSelect.addEventListener('change', renderStockCard);

// --- Initialization ---
document.addEventListener('DOMContentLoaded', () => {
  loadData(); // Load data when the page loads
  populateProductDropdowns(); // Populate product lists
  renderStockSummary(); // Display initial stock summary
  switchSection('dashboard'); // Default to dashboard view
});
```