<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cattle Management System</title>

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: 'Arial', sans-serif;

background: linear-gradient(135deg, #f0f8f0 0%, #e8f5e8 100%);

min-height: 100vh;

color: #2d5a2d;

}

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

.header {

background: linear-gradient(135deg, #2d5a2d 0%, #4a7c59 100%);

color: white;

padding: 30px 0;

text-align: center;

border-radius: 15px;

margin-bottom: 30px;

box-shadow: 0 8px 32px rgba(45, 90, 45, 0.3);

}

.header h1 {

font-size: 2.5em;

margin-bottom: 10px;

text-shadow: 2px 2px 4px rgba(0,0,0,0.3);

}

.header p {

font-size: 1.2em;

opacity: 0.9;

}

.card {

background: white;

border-radius: 15px;

padding: 25px;

margin-bottom: 25px;

box-shadow: 0 8px 32px rgba(45, 90, 45, 0.1);

border: 1px solid #e0f0e0;

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.card:hover {

transform: translateY(-2px);

box-shadow: 0 12px 40px rgba(45, 90, 45, 0.15);

}

.card h2 {

color: #2d5a2d;

margin-bottom: 20px;

font-size: 1.8em;

border-bottom: 3px solid #4a7c59;

padding-bottom: 10px;

}

.form-group {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 8px;

font-weight: bold;

color: #2d5a2d;

font-size: 1.1em;

}

input[type="number"], input[type="date"], select {

width: 100%;

padding: 15px;

border: 2px solid #d0e0d0;

border-radius: 10px;

font-size: 1.1em;

transition: border-color 0.3s ease;

background: #f9fdf9;

}

input[type="number"]:focus, input[type="date"]:focus, select:focus {

outline: none;

border-color: #4a7c59;

box-shadow: 0 0 0 3px rgba(74, 124, 89, 0.1);

}

.btn {

background: linear-gradient(135deg, #4a7c59 0%, #2d5a2d 100%);

color: white;

padding: 15px 30px;

border: none;

border-radius: 10px;

cursor: pointer;

font-size: 1.1em;

font-weight: bold;

transition: all 0.3s ease;

box-shadow: 0 4px 15px rgba(45, 90, 45, 0.3);

}

.btn:hover {

transform: translateY(-2px);

box-shadow: 0 8px 25px rgba(45, 90, 45, 0.4);

}

.btn:active {

transform: translateY(0);

}

.btn-secondary {

background: linear-gradient(135deg, #6c9478 0%, #5a7d66 100%);

margin-left: 10px;

}

.btn-danger {

background: linear-gradient(135deg, #c85a5a 0%, #a04545 100%);

}

.grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 20px;

margin-bottom: 25px;

}

.stats-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 20px;

margin-bottom: 25px;

}

.stat-card {

background: linear-gradient(135deg, #4a7c59 0%, #2d5a2d 100%);

color: white;

padding: 20px;

border-radius: 15px;

text-align: center;

box-shadow: 0 8px 32px rgba(45, 90, 45, 0.3);

}

.stat-card h3 {

font-size: 2em;

margin-bottom: 5px;

}

.stat-card p {

opacity: 0.9;

font-size: 1.1em;

}

.profit {

background: linear-gradient(135deg, #4a7c59 0%, #2d5a2d 100%);

}

.loss {

background: linear-gradient(135deg, #c85a5a 0%, #a04545 100%);

}

.chart-container {

position: relative;

height: 400px;

margin-bottom: 20px;

}

.tabs {

display: flex;

margin-bottom: 20px;

border-radius: 10px;

overflow: hidden;

box-shadow: 0 4px 15px rgba(45, 90, 45, 0.1);

}

.tab {

flex: 1;

padding: 15px;

background: #f0f8f0;

border: none;

cursor: pointer;

font-size: 1.1em;

font-weight: bold;

color: #2d5a2d;

transition: all 0.3s ease;

}

.tab.active {

background: linear-gradient(135deg, #4a7c59 0%, #2d5a2d 100%);

color: white;

}

.hidden {

display: none;

}

.alert {

padding: 15px;

margin-bottom: 20px;

border-radius: 10px;

font-weight: bold;

}

.alert-success {

background: #d4edda;

border: 1px solid #c3e6cb;

color: #155724;

}

.alert-danger {

background: #f8d7da;

border: 1px solid #f5c6cb;

color: #721c24;

}

.data-table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

.data-table th, .data-table td {

padding: 12px;

text-align: left;

border-bottom: 1px solid #e0f0e0;

}

.data-table th {

background: #f0f8f0;

color: #2d5a2d;

font-weight: bold;

}

.data-table tr:hover {

background: #f9fdf9;

}

@media (max-width: 768px) {

.container {

padding: 10px;

}

.header h1 {

font-size: 1.8em;

}

.grid {

grid-template-columns: 1fr;

}

.stats-grid {

grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

}

.tabs {

flex-direction: column;

}

.btn {

width: 100%;

margin: 5px 0;

}

}

</style>

</head>

<body>

<div class="container">

<div class="header">

<h1>🐄 Cattle Management System</h1>

<p>Track your milk production, costs, and profits easily</p>

</div>

<div id="alerts"></div>

<!-- Daily Milk Production Entry -->

<div class="card">

<h2>📊 Daily Milk Production</h2>

<div class="grid">

<div class="form-group">

<label for="date">Date:</label>

<input type="date" id="date" required>

</div>

<div class="form-group">

<label for="morning">Morning Milk (Liters):</label>

<input type="number" id="morning" step="0.1" min="0" placeholder="0.0">

</div>

<div class="form-group">

<label for="evening">Evening Milk (Liters):</label>

<input type="number" id="evening" step="0.1" min="0" placeholder="0.0">

</div>

</div>

<button class="btn" onclick="addMilkRecord()">Add Daily Record</button>

</div>

<!-- Cost Settings -->

<div class="card">

<h2>💰 Cost Settings</h2>

<div class="grid">

<div class="form-group">

<label for="milkPrice">Milk Price per Liter (₹):</label>

<input type="number" id="milkPrice" step="0.01" min="0" placeholder="0.00">

<button class="btn btn-secondary" onclick="updateMilkPrice()">Update Monthly Price</button>

</div>

<div class="form-group">

<label for="feedCost">Weekly Feed Cost (₹):</label>

<input type="number" id="feedCost" step="0.01" min="0" placeholder="0.00">

<button class="btn btn-secondary" onclick="updateFeedCost()">Update Weekly Cost</button>

</div>

</div>

</div>

<!-- Statistics Dashboard -->

<div class="stats-grid">

<div class="stat-card">

<h3 id="todayMilk">0.0</h3>

<p>Today's Milk (L)</p>

</div>

<div class="stat-card">

<h3 id="weekMilk">0.0</h3>

<p>This Week (L)</p>

</div>

<div class="stat-card">

<h3 id="monthMilk">0.0</h3>

<p>This Month (L)</p>

</div>

<div class="stat-card" id="profitCard">

<h3 id="monthProfit">₹0.00</h3>

<p>Monthly Profit/Loss</p>

</div>

</div>

<!-- Reports -->

<div class="card">

<h2>📈 Reports & Analytics</h2>

<div class="tabs">

<button class="tab active" onclick="showTab('weekly')">Weekly</button>

<button class="tab" onclick="showTab('monthly')">Monthly</button>

<button class="tab" onclick="showTab('yearly')">Yearly</button>

</div>

<div id="weekly" class="tab-content">

<div class="chart-container">

<canvas id="weeklyChart"></canvas>

</div>

<button class="btn" onclick="downloadWeeklyPDF()">Download PDF</button>

<button class="btn btn-secondary" onclick="downloadWeeklyExcel()">Download Excel</button>

</div>

<div id="monthly" class="tab-content hidden">

<div class="chart-container">

<canvas id="monthlyChart"></canvas>

</div>

<button class="btn" onclick="downloadMonthlyPDF()">Download PDF</button>

<button class="btn btn-secondary" onclick="downloadMonthlyExcel()">Download Excel</button>

</div>

<div id="yearly" class="tab-content hidden">

<div class="chart-container">

<canvas id="yearlyChart"></canvas>

</div>

<button class="btn" onclick="downloadYearlyPDF()">Download PDF</button>

<button class="btn btn-secondary" onclick="downloadYearlyExcel()">Download Excel</button>

</div>

</div>

<!-- Data Management -->

<div class="card">

<h2>🗂️ Data Management</h2>

<button class="btn btn-secondary" onclick="exportAllData()">Export All Data</button>

<button class="btn btn-danger" onclick="clearAllData()">Clear All Data</button>

</div>

<!-- Recent Records -->

<div class="card">

<h2>📋 Recent Records</h2>

<div id="recentRecords"></div>

</div>

</div>

<script>

// Data storage

let milkRecords = JSON.parse(localStorage.getItem('milkRecords')) || [];

let milkPrice = parseFloat(localStorage.getItem('milkPrice')) || 0;

let feedCost = parseFloat(localStorage.getItem('feedCost')) || 0;

// Charts

let weeklyChart, monthlyChart, yearlyChart;

// Initialize app

document.addEventListener('DOMContentLoaded', function() {

document.getElementById('date').valueAsDate = new Date();

document.getElementById('milkPrice').value = milkPrice;

document.getElementById('feedCost').value = feedCost;

updateDashboard();

initializeCharts();

displayRecentRecords();

});

// Add milk record

function addMilkRecord() {

const date = document.getElementById('date').value;

const morning = parseFloat(document.getElementById('morning').value) || 0;

const evening = parseFloat(document.getElementById('evening').value) || 0;

if (!date) {

showAlert('Please select a date', 'danger');

return;

}

if (morning === 0 && evening === 0) {

showAlert('Please enter milk production data', 'danger');

return;

}

const total = morning + evening;

const record = {

date: date,

morning: morning,

evening: evening,

total: total,

timestamp: new Date().toISOString()

};

// Check if record exists for this date

const existingIndex = milkRecords.findIndex(r => r.date === date);

if (existingIndex !== -1) {

milkRecords[existingIndex] = record;

showAlert('Record updated successfully!', 'success');

} else {

milkRecords.push(record);

showAlert('Record added successfully!', 'success');

}

localStorage.setItem('milkRecords', JSON.stringify(milkRecords));

// Clear form

document.getElementById('morning').value = '';

document.getElementById('evening').value = '';

updateDashboard();

updateCharts();

displayRecentRecords();

}

// Update milk price

function updateMilkPrice() {

const price = parseFloat(document.getElementById('milkPrice').value) || 0;

if (price <= 0) {

showAlert('Please enter a valid milk price', 'danger');

return;

}

milkPrice = price;

localStorage.setItem('milkPrice', milkPrice);

showAlert('Milk price updated successfully!', 'success');

updateDashboard();

}

// Update feed cost

function updateFeedCost() {

const cost = parseFloat(document.getElementById('feedCost').value) || 0;

if (cost <= 0) {

showAlert('Please enter a valid feed cost', 'danger');

return;

}

feedCost = cost;

localStorage.setItem('feedCost', feedCost);

showAlert('Feed cost updated successfully!', 'success');

updateDashboard();

}

// Update dashboard

function updateDashboard() {

const today = new Date().toISOString().split('T')[0];

const thisWeek = getWeekDates();

const thisMonth = getMonthDates();

// Today's milk

const todayRecord = milkRecords.find(r => r.date === today);

document.getElementById('todayMilk').textContent = todayRecord ? todayRecord.total.toFixed(1) : '0.0';

// This week's milk

const weekMilk = milkRecords

.filter(r => thisWeek.includes(r.date))

.reduce((sum, r) => sum + r.total, 0);

document.getElementById('weekMilk').textContent = weekMilk.toFixed(1);

// This month's milk

const monthMilk = milkRecords

.filter(r => thisMonth.includes(r.date))

.reduce((sum, r) => sum + r.total, 0);

document.getElementById('monthMilk').textContent = monthMilk.toFixed(1);

// Monthly profit/loss

const monthRevenue = monthMilk \* milkPrice;

const monthFeedCost = (feedCost \* 4.33); // Average weeks per month

const monthProfit = monthRevenue - monthFeedCost;

document.getElementById('monthProfit').textContent = '₹' + monthProfit.toFixed(2);

const profitCard = document.getElementById('profitCard');

if (monthProfit >= 0) {

profitCard.className = 'stat-card profit';

} else {

profitCard.className = 'stat-card loss';

}

}

// Show alert

function showAlert(message, type) {

const alertsDiv = document.getElementById('alerts');

const alert = document.createElement('div');

alert.className = `alert alert-${type}`;

alert.textContent = message;

alertsDiv.appendChild(alert);

setTimeout(() => {

alert.remove();

}, 3000);

}

// Get week dates

function getWeekDates() {

const today = new Date();

const week = [];

const startOfWeek = new Date(today);

startOfWeek.setDate(today.getDate() - today.getDay());

for (let i = 0; i < 7; i++) {

const date = new Date(startOfWeek);

date.setDate(startOfWeek.getDate() + i);

week.push(date.toISOString().split('T')[0]);

}

return week;

}

// Get month dates

function getMonthDates() {

const today = new Date();

const year = today.getFullYear();

const month = today.getMonth();

const daysInMonth = new Date(year, month + 1, 0).getDate();

const dates = [];

for (let day = 1; day <= daysInMonth; day++) {

const date = new Date(year, month, day);

dates.push(date.toISOString().split('T')[0]);

}

return dates;

}

// Initialize charts

function initializeCharts() {

// Weekly Chart

const weeklyCtx = document.getElementById('weeklyChart').getContext('2d');

weeklyChart = new Chart(weeklyCtx, {

type: 'line',

data: {

labels: [],

datasets: [{

label: 'Daily Milk Production (L)',

data: [],

borderColor: '#4a7c59',

backgroundColor: 'rgba(74, 124, 89, 0.1)',

tension: 0.4

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

plugins: {

title: {

display: true,

text: 'Weekly Milk Production'

}

}

}

});

// Monthly Chart

const monthlyCtx = document.getElementById('monthlyChart').getContext('2d');

monthlyChart = new Chart(monthlyCtx, {

type: 'bar',

data: {

labels: [],

datasets: [{

label: 'Weekly Milk Production (L)',

data: [],

backgroundColor: '#4a7c59',

borderColor: '#2d5a2d',

borderWidth: 1

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

plugins: {

title: {

display: true,

text: 'Monthly Milk Production'

}

}

}

});

// Yearly Chart

const yearlyCtx = document.getElementById('yearlyChart').getContext('2d');

yearlyChart = new Chart(yearlyCtx, {

type: 'line',

data: {

labels: [],

datasets: [{

label: 'Monthly Milk Production (L)',

data: [],

borderColor: '#4a7c59',

backgroundColor: 'rgba(74, 124, 89, 0.1)',

tension: 0.4

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

plugins: {

title: {

display: true,

text: 'Yearly Milk Production'

}

}

}

});

updateCharts();

}

// Update charts

function updateCharts() {

updateWeeklyChart();

updateMonthlyChart();

updateYearlyChart();

}

// Update weekly chart

function updateWeeklyChart() {

const weekDates = getWeekDates();

const weekData = weekDates.map(date => {

const record = milkRecords.find(r => r.date === date);

return record ? record.total : 0;

});

const weekLabels = weekDates.map(date => {

const d = new Date(date);

return d.toLocaleDateString('en-US', { weekday: 'short' });

});

weeklyChart.data.labels = weekLabels;

weeklyChart.data.datasets[0].data = weekData;

weeklyChart.update();

}

// Update monthly chart

function updateMonthlyChart() {

const today = new Date();

const weeks = [];

const weekData = [];

for (let i = 0; i < 4; i++) {

const weekStart = new Date(today);

weekStart.setDate(today.getDate() - today.getDay() - (i \* 7));

const weekEnd = new Date(weekStart);

weekEnd.setDate(weekStart.getDate() + 6);

weeks.unshift(`Week ${4 - i}`);

const weekMilk = milkRecords

.filter(r => {

const recordDate = new Date(r.date);

return recordDate >= weekStart && recordDate <= weekEnd;

})

.reduce((sum, r) => sum + r.total, 0);

weekData.unshift(weekMilk);

}

monthlyChart.data.labels = weeks;

monthlyChart.data.datasets[0].data = weekData;

monthlyChart.update();

}

// Update yearly chart

function updateYearlyChart() {

const today = new Date();

const months = [];

const monthData = [];

for (let i = 11; i >= 0; i--) {

const monthDate = new Date(today);

monthDate.setMonth(today.getMonth() - i);

const monthName = monthDate.toLocaleDateString('en-US', { month: 'short' });

months.push(monthName);

const monthMilk = milkRecords

.filter(r => {

const recordDate = new Date(r.date);

return recordDate.getMonth() === monthDate.getMonth() &&

recordDate.getFullYear() === monthDate.getFullYear();

})

.reduce((sum, r) => sum + r.total, 0);

monthData.push(monthMilk);

}

yearlyChart.data.labels = months;

yearlyChart.data.datasets[0].data = monthData;

yearlyChart.update();

}

// Show tab

function showTab(tabName) {

// Hide all tabs

document.querySelectorAll('.tab-content').forEach(tab => {

tab.classList.add('hidden');

});

// Remove active class from all tabs

document.querySelectorAll('.tab').forEach(tab => {

tab.classList.remove('active');

});

// Show selected tab

document.getElementById(tabName).classList.remove('hidden');

event.target.classList.add('active');

}

// Display recent records

function displayRecentRecords() {

const recentRecordsDiv = document.getElementById('recentRecords');

const recentRecords = milkRecords.slice(-10).reverse();

if (recentRecords.length === 0) {

recentRecordsDiv.innerHTML = '<p>No records found. Add your first milk production record!</p>';

return;

}

let html = '<table class="data-table"><thead><tr><th>Date</th><th>Morning (L)</th><th>Evening (L)</th><th>Total (L)</th></tr></thead><tbody>';

recentRecords.forEach(record => {

html += `<tr>

<td>${new Date(record.date).toLocaleDateString()}</td>

<td>${record.morning.toFixed(1)}</td>

<td>${record.evening.toFixed(1)}</td>

<td><strong>${record.total.toFixed(1)}</strong></td>

</tr>`;

});

html += '</tbody></table>';

recentRecordsDiv.innerHTML = html;

}

// Download functions

function downloadWeeklyPDF() {

const { jsPDF } = window.jspdf;

const doc = new jsPDF();

doc.setFontSize(18);

doc.text('Weekly Milk Production Report', 20, 20);

const weekDates = getWeekDates();

let yPosition = 40;

weekDates.forEach(date => {

const record = milkRecords.find(r => r.date === date);

const dateStr = new Date(date).toLocaleDateString();

const milk = record ? record.total.toFixed(1) : '0.0';

doc.text(`${dateStr}: ${milk}L`, 20, yPosition);

yPosition += 10;

});

doc.save('weekly-report.pdf');

}

function downloadWeeklyExcel() {

const weekDates = getWeekDates();

const data = [];

weekDates.forEach(date => {

const record = milkRecords.find(r => r.date === date);

data.push({

Date: new Date(date).toLocaleDateString(),

Morning: record ? record.morning : 0,

Evening: record ? record.evening : 0,

Total: record ? record.total : 0

});

});

const ws = XLSX.utils.json\_to\_sheet(data);

const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, 'Weekly Report');

XLSX.writeFile(wb, 'weekly-report.xlsx');

}

function downloadMonthlyPDF() {

const { jsPDF } = window.jspdf;

const doc = new jsPDF();

doc.setFontSize(18);

doc.text('Monthly Milk Production Report', 20, 20);

const monthDates = getMonthDates();

const totalMilk = milkRecords

.filter(r => monthDates.includes(r.date))

.reduce((sum, r) => sum + r.total, 0);

const revenue = totalMilk \* milkPrice;

const costs = feedCost \* 4.33;

const profit = revenue - costs;

doc.text(`Total Milk: ${totalMilk.toFixed(1)}L`, 20, 40);

doc.text(`Revenue: ₹${revenue.toFixed(2)}`, 20, 50);

doc.text(`Feed Cost: ₹${costs.toFixed(2)}`, 20, 60);

doc.text(`Profit/Loss: ₹${profit.toFixed(2)}`, 20, 70);

doc.save('monthly-report.pdf');

}

Function downloadMonthlyExcel() {

Const monthDates = getMonthDates();

Const data = [];

monthDates.forEach(date => {

const record = milkRecords.find(r => r.date === date);

if (record) {

data.push({

Date: new Date(date).toLocaleDateString(),

Morning: record.morning,

Evening: record.evening,

Total: record.total

});

}

});

// Add summary row

Const totalMilk = data.reduce((sum, row) => sum + row.Total, 0);

Const revenue = totalMilk \* milkPrice;

Const costs = feedCost \* 4.33;

Const profit = revenue – costs;

Data.push({

Date: ‘SUMMARY’,

Morning: ‘’,

Evening: ‘’,

Total: totalMilk.toFixed(1)

});

Data.push({

Date: ‘Revenue’,

Morning: ‘’,

Evening: ‘’,

Total: ‘₹’ + revenue.toFixed(2)

});

Data.push({

Date: ‘Feed Cost’,

Morning: ‘’,

Evening: ‘’,

Total: ‘₹’ + costs.toFixed(2)

});

Data.push({

Date: ‘Profit/Loss’,

Morning: ‘’,

Evening: ‘’,

Total: ‘₹’ + profit.toFixed(2)

});

Const ws = XLSX.utils.json\_to\_sheet(data);

Const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, ‘Monthly Report’);

XLSX.writeFile(wb, ‘monthly-report.xlsx’);

}

Function downloadYearlyPDF() {

Const { jsPDF } = window.jspdf;

Const doc = new jsPDF();

Doc.setFontSize(18);

Doc.text(‘Yearly Milk Production Report’, 20, 20);

Const today = new Date();

Let yPosition = 40;

Let totalYearlyMilk = 0;

// Generate monthly data for the year

For (let I = 11; I >= 0; i--) {

Const monthDate = new Date(today);

monthDate.setMonth(today.getMonth() – i);

const monthName = monthDate.toLocaleDateString(‘en-US’, { month: ‘long’, year: ‘numeric’ });

const monthMilk = milkRecords

.filter(r => {

Const recordDate = new Date(r.date);

Return recordDate.getMonth() === monthDate.getMonth() &&

recordDate.getFullYear() === monthDate.getFullYear();

})

.reduce((sum, r) => sum + r.total, 0);

totalYearlyMilk += monthMilk;

doc.text(`${monthName}: ${monthMilk.toFixed(1)}L`, 20, yPosition);

yPosition += 10;

if (yPosition > 250) {

doc.addPage();

yPosition = 20;

}

}

// Add summary

Const yearlyRevenue = totalYearlyMilk \* milkPrice;

Const yearlyCosts = feedCost \* 52; // 52 weeks per year

Const yearlyProfit = yearlyRevenue – yearlyCosts;

yPosition += 10;

doc.setFontSize(14);

doc.text(‘YEARLY SUMMARY’, 20, yPosition);

yPosition += 10;

doc.text(`Total Milk: ${totalYearlyMilk.toFixed(1)}L`, 20, yPosition);

yPosition += 10;

doc.text(`Revenue: ₹${yearlyRevenue.toFixed(2)}`, 20, yPosition);

yPosition += 10;

doc.text(`Feed Cost: ₹${yearlyCosts.toFixed(2)}`, 20, yPosition);

yPosition += 10;

doc.text(`Profit/Loss: ₹${yearlyProfit.toFixed(2)}`, 20, yPosition);

doc.save(‘yearly-report.pdf’);

}

Function downloadYearlyExcel() {

Const today = new Date();

Const data = [];

Let totalYearlyMilk = 0;

// Generate monthly data for the year

For (let I = 11; I >= 0; i--) {

Const monthDate = new Date(today);

monthDate.setMonth(today.getMonth() – i);

const monthName = monthDate.toLocaleDateString(‘en-US’, { month: ‘long’, year: ‘numeric’ });

const monthMilk = milkRecords

.filter(r => {

Const recordDate = new Date(r.date);

Return recordDate.getMonth() === monthDate.getMonth() &&

recordDate.getFullYear() === monthDate.getFullYear();

})

.reduce((sum, r) => sum + r.total, 0);

totalYearlyMilk += monthMilk;

data.push({

Month: monthName,

‘Milk Production (L)’: monthMilk.toFixed(1),

‘Revenue (₹)’: (monthMilk \* milkPrice).toFixed(2),

‘Feed Cost (₹)’: (feedCost \* 4.33).toFixed(2),

‘Profit/Loss (₹)’: ((monthMilk \* milkPrice) – (feedCost \* 4.33)).toFixed(2)

});

}

// Add yearly summary

Const yearlyRevenue = totalYearlyMilk \* milkPrice;

Const yearlyCosts = feedCost \* 52;

Const yearlyProfit = yearlyRevenue – yearlyCosts;

Data.push({

Month: ‘YEARLY TOTAL’,

‘Milk Production (L)’: totalYearlyMilk.toFixed(1),

‘Revenue (₹)’: yearlyRevenue.toFixed(2),

‘Feed Cost (₹)’: yearlyCosts.toFixed(2),

‘Profit/Loss (₹)’: yearlyProfit.toFixed(2)

});

Const ws = XLSX.utils.json\_to\_sheet(data);

Const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, ‘Yearly Report’);

XLSX.writeFile(wb, ‘yearly-report.xlsx’);

}

// Export all data

Function exportAllData() {

Const allData = {

milkRecords: milkRecords,

milkPrice: milkPrice,

feedCost: feedCost,

exportDate: new Date().toISOString()

};

// Create Excel file with all data

Const ws1 = XLSX.utils.json\_to\_sheet(milkRecords);

Const ws2 = XLSX.utils.json\_to\_sheet([{

‘Milk Price per Liter’: milkPrice,

‘Weekly Feed Cost’: feedCost,

‘Export Date’: new Date().toLocaleDateString()

}]);

Const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws1, ‘Milk Records’);

XLSX.utils.book\_append\_sheet(wb, ws2, ‘Settings’);

XLSX.writeFile(wb, ‘cattle-management-data.xlsx’);

// Also create JSON backup

Const jsonData = JSON.stringify(allData, null, 2);

Const blob = new Blob([jsonData], { type: ‘application/json’ });

Const url = URL.createObjectURL(blob);

Const a = document.createElement(‘a’);

a.href = url;

a.download = ‘cattle-management-backup.json’;

a.click();

URL.revokeObjectURL(url);

showAlert(‘All data exported successfully!’, ‘success’);

}

// Clear all data

Function clearAllData() {

If (confirm(‘Are you sure you want to clear all data? This action cannot be undone.’)) {

If (confirm(‘This will delete all your milk records, prices, and costs. Are you absolutely sure?’)) {

localStorage.removeItem(‘milkRecords’);

localStorage.removeItem(‘milkPrice’);

localStorage.removeItem(‘feedCost’);

milkRecords = [];

milkPrice = 0;

feedCost = 0;

document.getElementById(‘milkPrice’).value = 0;

document.getElementById(‘feedCost’).value = 0;

updateDashboard();

updateCharts();

displayRecentRecords();

showAlert(‘All data cleared successfully!’, ‘success’);

}

}

}

// Import data function (bonus feature)

Function importData() {

Const input = document.createElement(‘input’);

Input.type = ‘file’;

Input.accept = ‘.json’;

Input.onchange = function(event) {

Const file = event.target.files[0];

If (file) {

Const reader = new FileReader();

Reader.onload = function€ {

Try {

Const data = JSON.parse(e.target.result);

If (data.milkRecords && Array.isArray(data.milkRecords)) {

milkRecords = data.milkRecords;

localStorage.setItem(‘milkRecords’, JSON.stringify(milkRecords));

}

If (data.milkPrice && typeof data.milkPrice === ‘number’) {

milkPrice = data.milkPrice;

localStorage.setItem(‘milkPrice’, milkPrice);

document.getElementById(‘milkPrice’).value = milkPrice;

}

If (data.feedCost && typeof data.feedCost === ‘number’) {

feedCost = data.feedCost;

localStorage.setItem(‘feedCost’, feedCost);

document.getElementById(‘feedCost’).value = feedCost;

}

updateDashboard();

updateCharts();

displayRecentRecords();

showAlert(‘Data imported successfully!’, ‘success’);

} catch (error) {

showAlert(‘Error importing data. Please check the file format.’, ‘danger’);

}

};

Reader.readAsText(file);

}

};

Input.click();

}

// Auto-save functionality

setInterval(function() {

if (milkRecords.length > 0) {

localStorage.setItem(‘milkRecords’, JSON.stringify(milkRecords));

}

}, 30000); // Auto-save every 30 seconds

// Add keyboard shortcuts

Document.addEventListener(‘keydown’, function€ {

// Ctrl + S to save/export data

If (e.ctrlKey && e.key === ‘s’) {

e.preventDefault();

exportAllData();

}

// Ctrl + N to add new record

If (e.ctrlKey && e.key === ‘n’) {

e.preventDefault();

document.getElementById(‘morning’).focus();

}

});

// PWA Service Worker Registration (for offline functionality)

If (‘serviceWorker’ in navigator) {

Window.addEventListener(‘load’, function() {

Navigator.serviceWorker.register(‘/sw.js’).then(function(registration) {

Console.log(‘ServiceWorker registration successful’);

}, function(err) {

Console.log(‘ServiceWorker registration failed: ‘, err);

});

});

}

// Add print functionality

Function printReport(reportType) {

Const printWindow = window.open(‘’, ‘\_blank’);

Const currentDate = new Date().toLocaleDateString();

Let content = `

<html>

<head>

<title>${reportType} Report – Cattle Management</title>

<style>

Body { font-family: Arial, sans-serif; margin: 20px; }

H1 { color: #2d5a2d; text-align: center; }

Table { width: 100%; border-collapse: collapse; margin: 20px 0; }

Th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }

Th { background-color: #f0f8f0; }

.summary { background-color: #e8f5e8; font-weight: bold; }

</style>

</head>

<body>

<h1>${reportType} Milk Production Report</h1>

<p>Generated on: ${currentDate}</p>

`;

// Add specific report content based on type

If (reportType === ‘Weekly’) {

Const weekDates = getWeekDates();

Content += ‘<table><tr><th>Date</th><th>Morning (L)</th><th>Evening (L)</th><th>Total (L)</th></tr>’;

Let weekTotal = 0;

weekDates.forEach(date => {

const record = milkRecords.find(r => r.date === date);

const morning = record ? record.morning : 0;

const evening = record ? record.evening : 0;

const total = record ? record.total : 0;

weekTotal += total;

content += `<tr>

<td>${new Date(date).toLocaleDateString()}</td>

<td>${morning.toFixed(1)}</td>

<td>${evening.toFixed(1)}</td>

<td>${total.toFixed(1)}</td>

</tr>`;

});

Content += `<tr class=”summary”>

<td>TOTAL</td>

<td>-</td>

<td>-</td>

<td>${weekTotal.toFixed(1)}</td>

</tr>`;

}

Content += ‘</table></body></html>’;

printWindow.document.write(content);

printWindow.document.close();

printWindow.print();

}

// Add data validation

Function validateMilkData(morning, evening) {

If (morning < 0 || evening < 0) {

Return “Milk production cannot be negative”;

}

If (morning > 100 || evening > 100) {

Return “Milk production seems unusually high. Please verify.”;

}

Return null;

}

// Enhanced add milk record with validation

Function addMilkRecord() {

Const date = document.getElementById(‘date’).value;

Const morning = parseFloat(document.getElementById(‘morning’).value) || 0;

Const evening = parseFloat(document.getElementById(‘evening’).value) || 0;

If (!date) {

showAlert(‘Please select a date’, ‘danger’);

return;

}

If (morning === 0 && evening === 0) {

showAlert(‘Please enter milk production data’, ‘danger’);

return;

}

// Validate data

Const validationError = validateMilkData(morning, evening);

If (validationError) {

showAlert(validationError, ‘danger’);

return;

}

Const total = morning + evening;

Const record = {

Date: date,

Morning: morning,

Evening: evening,

Total: total,

Timestamp: new Date().toISOString()

};

// Check if record exists for this date

Const existingIndex = milkRecords.findIndex(r => r.date === date);

If (existingIndex !== -1) {

milkRecords[existingIndex] = record;

showAlert(‘Record updated successfully!’, ‘success’);

} else {

milkRecords.push(record);

showAlert(‘Record added successfully!’, ‘success’);

}

localStorage.setItem(‘milkRecords’, JSON.stringify(milkRecords));

// Clear form

Document.getElementById(‘morning’).value = ‘’;

Document.getElementById(‘evening’).value = ‘’;

updateDashboard();

updateCharts();

displayRecentRecords();

}

// Initialize app when DOM is loaded

Document.addEventListener(‘DOMContentLoaded’, function() {

Document.getElementById(‘date’).valueAsDate = new Date();

Document.getElementById(‘milkPrice’).value = milkPrice;

Document.getElementById(‘feedCost’).value = feedCost;

updateDashboard();

initializeCharts();

displayRecentRecords();

// Add tooltips for better user experience

addTooltips();

});

// Add tooltips function

Function addTooltips() {

Const tooltips = {

‘morning’: ‘Enter morning milk production in liters’,

‘evening’: ‘Enter evening milk production in liters’,

‘milkPrice’: ‘Set the selling price per liter of milk’,

‘feedCost’: ‘Set the total weekly feed cost for cattle’

};

Object.keys(tooltips).forEach(id => {

Const element = document.getElementById(id);

If (element) {

Element.title = tooltips[id];

}

});

}

Console.log(‘Cattle Management System loaded successfully!’);