📦 Complete Deployment Guide - Inventory Scanner System

Table of Contents

System Requirements

Development Machine

- **Node.js**: v16.0 or higher
- **npm**: v7.0 or higher
- **Git**: Latest version
- **OS**: Windows 10/11, macOS 10.15+, or Ubuntu 20.04+
- **RAM**: Minimum 8GB
- **Storage**: 10GB free space

Mobile Development

- **Expo CLI**: Latest version
- **Android Studio** (for Android) or **Xcode** (for iOS)
- **Physical phone** for testing (recommended)

Part 1: Airtable Setup

Step 1.1: Create Airtable Account

1. Go to Airtable.com
2. Sign up for free account
3. Verify your email

Step 1.2: Create Base Structure

1. Click **"Start from scratch"**
2. Name your base: **"Inventory Management System"**
3. Delete the default table
4. Create 4 new tables (click + icon):
   - Users
   - Products
   - ScanHistory
   - Settings

Step 1.3: Configure Users Table

Click on "Users" table and add these fields (click + to add field):

```
Email (Email) - Already exists as primary field
Password (Single line text)
Name (Single line text)
Role (Single select) → Options: admin, manager, staff
TotalScans (Number) → Integer, Default: 0
CreatedAt (Date & time) → Include time field
LastActive (Date & time) → Include time field
```

Step 1.4: Configure Products Table

```
Barcode (Single line text) - Primary field
Name (Single line text)
Brand (Single line text)
Category (Single select) → Add options: Electronics, Clothing, Home & Garden, etc.
Description (Long text)
Price (Currency) → USD, Precision: 2
Cost (Currency) → USD, Precision: 2
Weight (Single line text)
Dimensions (Single line text)
Quantity (Number) → Integer, Default: 1
Location (Single select) → Add: A1, A2, B1, B2, etc.
Condition (Single select) → Options: new, like-new, good, fair, poor
Status (Single select) → Options: pending, listed, sold, returned
Images (Attachment)
Source (Single line text)
ApiData (Long text)
ScannedBy (Link to another record) → Link to Users table
ScannedByEmail (Single line text)
ScannedAt (Date & time)
LastModified (Date & time)
Notes (Long text)
```

Step 1.5: Configure ScanHistory Table

```
Timestamp (Date & time) - Primary field
User (Link to another record) → Link to Users
UserEmail (Single line text)
Product (Link to another record) → Link to Products
ProductName (Single line text)
Barcode (Single line text)
Action (Single select) → Options: scan, manual, update, delete
```

Step 1.6: Configure Settings Table

```
User (Link to another record) - Primary field, Link to Users
UserEmail (Email)
ApiProvider (Single select) → Options: openFood, upcItemDB, barcodeLookup
ScanSound (Checkbox)
AutoSave (Checkbox)
DefaultCondition (Single select) → Same as Products Condition
DefaultStatus (Single select) → Same as Products Status
UpdatedAt (Date & time)
```

Step 1.7: Get API Credentials

1. Go to https://airtable.com/create/tokens
2. Click **"Create new token"**
3. Name it: "Inventory Scanner API"
4. Add scopes:
   - `data.records:read`
   - `data.records:write`
   - `schema.bases:read`
5. Add your base under "Access"
6. Click **"Create token"**
7. Copy and save the token (starts with `pat`)

Step 1.8: Get Base ID

1. Go to https://airtable.com/api
2. Select your "Inventory Management System" base
3. Copy the Base ID (starts with `app`)

Part 2: Backend Deployment

Step 2.1: Setup Project Directory

```bash
```

```
# Create project folder
mkdir liquidation-inventory-system
cd liquidation-inventory-system

# Create backend folder
mkdir backend
cd backend

# Initialize npm project
npm init -y
```

Step 2.2: Install Dependencies

```bash
# Install all required packages
npm install express cors airtable bcryptjs jsonwebtoken axios dotenv multer

# Install development dependencies
npm install -D nodemon
```

Step 2.3: Create Project Structure

```bash
# Create necessary directories
mkdir uploads
mkdir config
mkdir routes
mkdir middleware
```

Step 2.4: Create Main Server File

Create `server.js`:

```javascript
// Copy the entire backend code from the artifact
// Save it as server.js in the backend folder
```

Step 2.5: Create Environment Configuration

Create `.env` file in backend folder:

```env
# Server Configuration
PORT=3000
NODE_ENV=development

# JWT Secret (Generate a random string)
JWT_SECRET=your-super-secret-jwt-key-change-this-123456789

# Airtable Configuration (REQUIRED - Use your actual credentials)
AIRTABLE_API_KEY=patXXXXXXXXXXXXXX
AIRTABLE_BASE_ID=appXXXXXXXXXXXXXX

# Product Lookup APIs (Optional but recommended)
UPC_ITEMDB_KEY=
BARCODE_LOOKUP_KEY=
```

Step 2.6: Update package.json

Edit `package.json`:

```json
```

```
{
  "name": "inventory-scanner-backend",
  "version": "1.0.0",
  "description": "Backend for liquidation inventory scanner",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "test": "node test-connection.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "cors": "^2.8.5",
    "airtable": "^0.12.2",
    "bcryptjs": "^2.4.3",
    "jsonwebtoken": "^9.0.0",
    "axios": "^1.4.0",
    "dotenv": "^16.0.3",
    "multer": "^1.4.5-lts.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}
```

Step 2.7: Test Backend Connection

Create `test-connection.js`:

```
javascript
```

```
require('dotenv').config();
const Airtable = require('airtable');

const base = new Airtable({ apiKey: process.env.AIRTABLE_API_KEY })
  .base(process.env.AIRTABLE_BASE_ID);

console.log('Testing Airtable connection...');

base('Users').select({ maxRecords: 1 }).firstPage((err, records) => {
  if (err) {
    console.error('❌ Connection failed:', err);
    return;
  }
  console.log('✅ Successfully connected to Airtable!');
  console.log('Found', records.length, 'user records');
});
```

Run test:

```bash
node test-connection.js
```

Step 2.8: Start Backend Server

```bash
# Development mode
npm run dev

# Production mode
npm start
```

You should see:

```
Server running on port 3000
Connected to Airtable
```

Part 3: Mobile App Deployment

Step 3.1: Install Expo CLI

```bash
# Install Expo CLI globally
npm install -g expo-cli

# Verify installation
expo --version
```

Step 3.2: Create Mobile App

```bash
# Go back to main project folder
cd ..

# Create new Expo project
expo init mobile-app

# Choose: blank (TypeScript) or blank (JavaScript)
# Name: InventoryScanner

cd mobile-app
```

Step 3.3: Install Required Dependencies

```bash
# Install all necessary packages
npm install @react-native-async-storage/async-storage
npm install axios
npm install expo-camera
npm install expo-barcode-scanner
npm install expo-image-picker
npm install expo-linear-gradient
```

Step 3.4: Configure App.js

Replace the default `App.js` with the mobile app code from the artifact.

Step 3.5: Update API Configuration

Edit `App.js` and update the API URL:

```javascript
```

```
// For local development (find your IP address)
const API_URL = 'http://192.168.1.100:3000/api';

// Windows: Run 'ipconfig' in command prompt
// Mac/Linux: Run 'ifconfig' in terminal
// Look for your IPv4 address
```

Step 3.6: Configure app.json

Update `app.json`:

```
json
```

```
{
  "expo": {
    "name": "Inventory Scanner Pro",
    "slug": "inventory-scanner",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "light",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#667eea"
    },
    "assetBundlePatterns": [
      "**/*"
    ],
    "ios": {
      "supportsTablet": true,
      "bundleIdentifier": "com.yourbusiness.inventoryscanner",
      "infoPlist": {
        "NSCameraUsageDescription": "This app needs camera access to scan barcodes",
        "NSPhotoLibraryUsageDescription": "This app needs photo library access to save product images"
      }
    },
    "android": {
      "adaptiveIcon": {
        "foregroundImage": "./assets/adaptive-icon.png",
        "backgroundColor": "#667eea"
      },
      "package": "com.yourbusiness.inventoryscanner",
      "permissions": [
        "CAMERA",
        "READ_EXTERNAL_STORAGE",
        "WRITE_EXTERNAL_STORAGE"
      ]
    }
  }
}
```

Step 3.7: Start Mobile App

```
bash
```

```
# Start Expo development server
expo start

# Options:
# Press 'a' for Android emulator
# Press 'i' for iOS simulator
# Scan QR code with Expo Go app on phone
```

Part 4: Testing & Verification

Step 4.1: Create Test User

Using the mobile app or API:

```bash
# Using curl to create test user
curl -X POST http://localhost:3000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "password": "password123",
    "name": "Test User"
  }'
```

Step 4.2: Test Barcode Scanning

Test barcodes:

- **Food**: `012000001086` (Pepsi)
- **Electronics**: `885909560739` (Apple product)
- **Books**: `9780307476463`
- **Generic**: `049000042566` (Coca-Cola)

Step 4.3: Verify Airtable Data

1. Go to your Airtable base
2. Check Products table for scanned items
3. Check ScanHistory for activity logs
4. Verify Users table shows correct scan counts

Step 4.4: Test All Features

- [ ] User registration
- [ ] User login
- [ ] Camera barcode scanning
- [ ] Manual barcode entry
- [ ] Product lookup from APIs
- [ ] Save product to Airtable
- [ ] View products list
- [ ] View scan history
- [ ] Export to CSV
- [ ] Settings update

Part 5: Production Deployment

Option A: Deploy to Cloud (Recommended)

Backend on Heroku

```bash
cd backend

# Install Heroku CLI
# Create Heroku app
heroku create your-inventory-api

# Set environment variables
heroku config:set JWT_SECRET=your-production-secret
heroku config:set AIRTABLE_API_KEY=your-api-key
heroku config:set AIRTABLE_BASE_ID=your-base-id

# Create Procfile
echo "web: node server.js" > Procfile

# Deploy
git init
git add .
git commit -m "Initial deployment"
git push heroku main
```

Backend on Railway.app (Easier Alternative)

1. Go to Railway.app
2. Connect GitHub repo
3. Add environment variables
4. Deploy automatically

Mobile App - Build APK/IPA

```bash
# Install EAS CLI
npm install -g eas-cli

# Login to Expo account
eas login

# Configure build
eas build:configure

# Build Android APK
eas build --platform android --profile preview

# Build iOS (requires Apple Developer Account)
eas build --platform ios --profile preview
```

Option B: Deploy On-Premise

Server Setup (Ubuntu 20.04)

```bash
```

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Install PM2 for process management
sudo npm install -g pm2

# Install Nginx for reverse proxy
sudo apt install nginx

# Clone your repository
git clone your-repo-url
cd your-repo/backend

# Install dependencies
npm install

# Start with PM2
pm2 start server.js --name inventory-api
pm2 save
pm2 startup
```

Configure Nginx

Create `/etc/nginx/sites-available/inventory`:

```nginx
```

```
server {
  listen 80;
  server_name your-domain.com;

  location / {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
  }
}
```

Enable site:

```bash
sudo ln -s /etc/nginx/sites-available/inventory /etc/nginx/sites-enabled
sudo nginx -t
sudo systemctl restart nginx
```

SSL Certificate (HTTPS)

```bash
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get SSL certificate
sudo certbot --nginx -d your-domain.com
```

Part 6: Troubleshooting

Common Issues & Solutions

1. Airtable Connection Failed

Error: "Invalid API Key" or "Base not found"

Solution:

```bash
# Check .env file for extra spaces
cat .env | grep AIRTABLE

# Test with curl
curl https://api.airtable.com/v0/YOUR_BASE_ID/Products \
  -H "Authorization: Bearer YOUR_API_KEY"
```

2. Mobile App Can't Connect to Backend

Error: "Network request failed"

Solution:

```javascript
// Check API URL in App.js
// For local development, use your computer's IP, not localhost
const API_URL = 'http://192.168.1.100:3000/api'; // Replace with your IP

// For production
const API_URL = 'https://your-api-domain.com/api';
```

3. Camera Not Working

Error: "Camera permission denied"

Solution:

- iOS: Settings → Privacy → Camera → Enable for Expo Go
- Android: Settings → Apps → Expo Go → Permissions → Camera

4. Build Failing

Error: Various build errors

Solution:

```bash
```

```
# Clear cache
expo start -c

# Reset Metro bundler
npx react-native start --reset-cache

# Reinstall dependencies
rm -rf node_modules
npm install
```

5. Airtable Rate Limiting

Error: "Rate limit exceeded"

Solution: Add delay between requests:

```javascript
// Add to backend
const delay = ms => new Promise(resolve => setTimeout(resolve, ms));

// Use in bulk operations
await delay(200); // 200ms delay = 5 requests per second
```

Debug Commands

```bash
# Check backend logs
pm2 logs inventory-api

# Monitor backend
pm2 monit

# Check Nginx logs
sudo tail -f /var/log/nginx/error.log

# Test API endpoint
curl http://localhost:3000/api/products \
  -H "Authorization: Bearer YOUR_TOKEN"

# Check port usage
sudo netstat -tlnp | grep 3000
```

Post-Deployment Checklist

Security

- [ ] Change default JWT secret
- [ ] Enable HTTPS
- [ ] Set up firewall rules
- [ ] Regular security updates
- [ ] Implement rate limiting

Monitoring

- [ ] Set up uptime monitoring (UptimeRobot)
- [ ] Configure error tracking (Sentry)
- [ ] Set up logging (LogDNA or similar)
- [ ] Monitor Airtable usage

Backup

- [ ] Daily Airtable snapshots
- [ ] Weekly CSV exports
- [ ] Document backup procedures

Performance

- [ ] Optimize image sizes
- [ ] Implement caching
- [ ] Monitor API response times
- [ ] Load testing

Documentation

- [ ] Create user manual
- [ ] Document API endpoints
- [ ] Training videos for staff
- [ ] FAQ section

Support & Maintenance

Daily Tasks

- Check system status

- Review error logs
- Monitor Airtable records

Weekly Tasks

- Export data backup
- Review scan statistics
- Update product catalog

Monthly Tasks

- Security updates
- Performance review
- User feedback collection
- Cost analysis

Getting Help

1. **Airtable Issues**: support@airtable.com
2. **Expo/React Native**: forums.expo.dev
3. **Node.js/Express**: stackoverflow.com
4. **API Services**: Check respective documentation

API Credentials Setup

Free API Options

Open Food Facts (Free, No Key Required)

- Website: https://world.openfoodfacts.org
- Rate Limit: 100 requests/minute
- Coverage: Food products globally

UPC ItemDB (Free Tier Available)

1. Register at https://www.upcitemdb.com
2. Get API key from dashboard
3. Free: 100 requests/day
4. Paid: From $29/month

Barcode Lookup (Premium)

1. Register at https://www.barcodelookup.com
2. Plans from $39-299/month
3. Best coverage for retail products
4. Add to .env: `BARCODE_LOOKUP_KEY=your-key`

Congratulations! 🎉

Your Inventory Scanner System is now deployed and ready for use.

Next Steps:

1. Train your staff on the mobile app
2. Import existing inventory (if any)
3. Set up automated reports in Airtable
4. Configure integrations (eBay, Amazon, etc.)

Need Additional Features?

- Bulk import/export tools
- Advanced analytics dashboard
- Multi-warehouse support
- Customer management system
- Integration with accounting software

For support, refer to the documentation or contact your system administrator.