JunkShop - Complete Setup & Deployment Guide

Last Updated: October 17, 2025
Project: JunkShop E-commerce Website
Repository: https://github.com/landsendsolo/junkshop_live

Table of Contents

1. Prerequisites
2. SumUp Account Setup
3. Local Development Setup
4. Firebase Configuration
5. SumUp Integration Configuration
6. Deployment to Production
7. Testing the Payment Flow
8. Troubleshooting
9. Post-Deployment Configuration

Prerequisites

Required Software

- **Node.js** 18 or higher
  `bash`
  ```
  node --version  # Should show v18.x.x or higher
  ```
- **Firebase CLI**
  `bash`
  ```
  npm install -g firebase-tools
  firebase --version
  ```
- **Git**
  `bash`
  ```
  git --version
  ```

Required Accounts

1. **Firebase Account** (Google account)
   - Project already created: `junkshop-website-gem`
2. **SumUp Merchant Account**
   - Sign up at: https://sumup.com/
   - Verify your business details
   - Complete merchant onboarding

SumUp Account Setup

Step 1: Create SumUp Developer Account

1. Log into your SumUp merchant account: https://me.sumup.com
2. Navigate to **Settings** → **Developer Settings** or **API Keys**
3. Generate a new API key

Step 2: Get Your API Credentials

Option A: API Key (Recommended for Getting Started)

1. In SumUp dashboard, go to Developer Settings
2. Click “Generate New API Key”
3. Copy the API key immediately (it won’t be shown again)
4. Save it securely

Option B: OAuth Application (For Production)

1. Navigate to OAuth Apps section
2. Create a new OAuth application:
   - **Application Name**: JunkShop E-commerce
   - **Homepage URL**: https://junkshop-website-gem.web.app
   - **Redirect URI**: https://junkshop-website-gem.web.app/oauth-callback
3. Note down:
   - Client ID
   - Client Secret

Step 3: Enable Test Mode (For Development)

1. In SumUp dashboard, switch to **Test Mode** or **Sandbox**
2. Generate test API credentials
3. Use these for all development and testing

Local Development Setup

Step 1: Clone the Repository

```
git clone https://github.com/landsendsolo/junkshop_live.git
cd junkshop_live
```

Step 2: Install Dependencies

```
# Install Firebase Functions dependencies
cd functions
npm install
cd ..
```

Step 3: Configure Environment Variables

```
# Create .env file from example
cp .env.example .env

# Edit .env and add your credentials
nano .env  # or use your preferred editor
```

Add your SumUp credentials:

```
SUMUP_API_KEY=your_test_api_key_here
```

Step 4: Login to Firebase

```
firebase login
```

Follow the prompts to authenticate with your Google account.

Step 5: Select Firebase Project

```
firebase use junkshop-website-gem
```

Step 6: Start Firebase Emulators

```
firebase emulators:start
```

This will start:
- **Hosting** at http://localhost:5000
- **Functions** at http://localhost:5001
- **Firestore** at http://localhost:8080

Step 7: Access Local Site

Open your browser and navigate to:
- **Main Site**: http://localhost:5000
- **Admin Panel**: http://localhost:5000/admin.html
- **Emulator UI**: http://localhost:4000

Firebase Configuration

Step 1: Verify Firebase Project Settings

```
firebase projects:list
```

Ensure you're using the correct project:

```
firebase use junkshop-website-gem
```

Step 2: Configure Authentication

1. Go to Firebase Console: https://console.firebase.google.com
2. Select your project: `junkshop-website-gem`
3. Navigate to **Authentication** → **Sign-in method**
4. Enable:
   - ✅ **Anonymous** (for customer browsing)
   - ✅ **Email/Password** (for admin access)

Step 3: Create Admin User

1. In Firebase Console → **Authentication** → **Users**
2. Click “Add User”
3. Enter:
   - **Email**: your-admin-email@example.com
   - **Password**: [secure password]
4. Save the credentials

Step 4: Configure Firestore Security Rules

The project should already have security rules, but verify:

1. Firebase Console → **Firestore Database** → **Rules**
2. Ensure rules allow:
   - Public read access to active products
   - Authenticated write access for orders
   - Admin write access for product management

Example rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Products: Public read, admin write
    match /products/{productId} {
      allow read: if true;
      allow write: if request.auth != null;
    }

    // Orders: Authenticated users can create, admins can read/update
    match /orders/{orderId} {
      allow create: if request.auth != null;
      allow read, update: if request.auth != null;
    }
  }
}
```

SumUp Integration Configuration

For Local Development (Using Emulators)

```
# Set environment variable
export SUMUP_API_KEY="your_test_api_key"

# Or add to .env file
echo "SUMUP_API_KEY=your_test_api_key" >> .env
```

For Firebase Cloud Functions (Production)

```
# Set Firebase Functions config
firebase functions:config:set sumup.api_key="YOUR_PRODUCTION_API_KEY"

# Verify configuration
firebase functions:config:get
```

Expected output:

```
{
  "sumup": {
    "api_key": "YOUR_API_KEY"
  }
}
```

Alternative: Using OAuth Access Token

```
firebase functions:config:set sumup.access_token="YOUR_ACCESS_TOKEN"
```

Deployment to Production

Pre-Deployment Checklist

- [ ] SumUp API credentials configured
- [ ] Firebase authentication enabled (Anonymous + Email/Password)
- [ ] Admin user created
- [ ] Firestore security rules reviewed
- [ ] Functions dependencies installed
- [ ] Test payment completed in test mode
- [ ] All files synced (root → public folder)

Step 1: Sync Files to Public Folder

```
# Ensure public folder has latest files
cp index.html public/index.html
cp admin.html public/admin.html
cp payment-success.html public/payment-success.html
```

Step 2: Deploy All Services

```
# Deploy everything (hosting + functions)
firebase deploy
```

Or deploy individually:

```
# Deploy only functions
firebase deploy --only functions

# Deploy only hosting
firebase deploy --only hosting

# Deploy only Firestore rules
firebase deploy --only firestore:rules
```

Step 3: Verify Deployment

After deployment completes, you'll see:

```
✔  Deploy complete!

Project Console: https://console.firebase.google.com/project/junkshop-website-gem/
overview
Hosting URL: https://junkshop-website-gem.web.app
```

Step 4: Test Live Site

1. Open: https://junkshop-website-gem.web.app
2. Verify homepage loads
3. Test product browsing
4. Add items to cart
5. Proceed to checkout (use test mode)

Testing the Payment Flow

Using SumUp Test Environment

1. **Enable Test Mode** in your SumUp account
2. **Use Test API Key** in Firebase config
3. **Test Card Numbers** (check SumUp documentation for current test cards)

Complete Test Checkout

1. **Add Products to Cart**
   - Browse products
   - Click "Add to Cart"
   - Verify cart updates

2. **Proceed to Checkout**
   - Click cart icon
   - Fill in customer details:
     - Name: Test Customer
     - Email: test@example.com
     - Address: 123 Test St, Test City, TE5T 123
3. **Initiate Payment**
   - Click “Pay with Card”
   - Verify redirect to SumUp hosted checkout page
   - Should see SumUp payment form (not JSON!)
4. **Complete Payment**
   - Enter test card details
   - Complete payment
   - Should redirect back to payment-success.html
5. **Verify Order Created**
   - Check Firebase Console → Firestore → orders collection
   - Order should have:
     - `status: "pending_payment"` initially
     - `status: "paid"` after webhook (may take a few seconds)
     - Products should be marked as `status: "sold"`
6. **Check Admin Panel**
   - Login to admin.html
   - Verify order appears in orders list
   - Check order status

Expected Flow Timeline

```
0:00 - Customer clicks "Pay with Card"
0:01 - Order saved with status "pending_payment"
0:02 - Redirect to SumUp hosted checkout
0:10 - Customer completes payment on SumUp
0:11 - SumUp processes payment
0:12 - Customer redirected to payment-success.html
0:13 - Success page shows "Payment Successful"
0:14 - SumUp webhook fires to backend
0:15 - Order status updated to "paid"
0:15 - Products marked as "sold"
```

Troubleshooting

Issue: Payment redirect shows JSON instead of payment form

Problem: Frontend redirecting to API endpoint instead of hosted checkout URL.

Solution:
- Ensure Cloud Function returns `hostedCheckoutUrl`

- Verify frontend uses `result.data.hostedCheckoutUrl` not `checkoutId`
- Check Cloud Function logs: `firebase functions:log`

Issue: "Payment system not configured" error

Problem: SumUp API key not set in Firebase Functions.

Solution:

```
# Check current config
firebase functions:config:get

# Set API key
firebase functions:config:set sumup.api_key="YOUR_KEY"

# Redeploy functions
firebase deploy --only functions
```

Issue: "Merchant code not available" error

Problem: SumUp API not returning merchant profile.

Solution:
- Verify API key is correct and active
- Check SumUp account is fully activated
- Ensure merchant account has completed onboarding
- Try generating a new API key

Issue: Webhook not updating order status

Problem: Order stays as "pending_payment" after successful payment.

Solution:
1. **Configure Webhook URL in SumUp**:
- Go to SumUp dashboard → Developer Settings
- Set webhook URL to:
`https://europe-west2-junkshop-website-gem.cloudfunctions.net/handleSumupWebhook`
- Enable events: `checkout.paid`, `checkout.failed`

1. **Check Function Logs**:
   `bash`
   `  firebase functions:log --only handleSumupWebhook`

2. **Verify Function Deployed**:
   `bash`
   `  firebase deploy --only functions:handleSumupWebhook`

Issue: Authentication errors

Problem: Anonymous sign-in failed.

Solution:
1. Firebase Console → Authentication → Sign-in method
2. Enable "Anonymous" provider
3. Refresh website

Issue: Products not loading

Problem: Firestore connection issues.

Solution:
1. Check browser console for errors
2. Verify Firestore security rules allow public read
3. Check Firebase project ID in config is correct

Issue: Admin login not working

Problem: Email/password authentication not configured.

Solution:
1. Enable Email/Password in Firebase Auth
2. Create admin user in Firebase Console
3. Use exact credentials to login

Post-Deployment Configuration

Configure SumUp Webhook

1. **Get Cloud Function URL**:
   `bash`
   `firebase functions:list`

Copy the URL for `handleSumupWebhook`, e.g.:
`https://europe-west2-junkshop-website-gem.cloudfunctions.net/handleSumupWebhook`

1. **Configure in SumUp Dashboard**:
   - Login to SumUp merchant account
   - Navigate to Developer Settings → Webhooks
   - Add new webhook:

     - **URL**: [Your Cloud Function URL]
     - **Events**:
     - ✅ checkout.paid
     - ✅ checkout.failed
     - ✅ checkout.status.updated
     - Save webhook configuration

2. **Test Webhook**:
   - Complete a test payment
   - Check Function logs:
   `bash`
   `firebase functions:log`
   - Verify order status updates

Enable Analytics (Optional)

1. Firebase Console → Analytics
2. Enable Google Analytics 4
3. Note your Measurement ID (already in config: `G-DL2J14LQP0`)

Set Up Backup Strategy

1. **Automated Firestore Backups**:
   - Firebase Console → Firestore → Import/Export
   - Schedule regular exports to Cloud Storage

2. **Function Backups**:
   - Code is in Git (already backed up)
   - Config: `firebase functions:config:get > config-backup.json`

Environment Variables Reference

Required for Production

```
# SumUp Integration
firebase functions:config:set sumup.api_key="YOUR_SUMUP_API_KEY"
```

Optional for Enhanced Features

```
# Email notifications (future enhancement)
firebase functions:config:set sendgrid.api_key="YOUR_SENDGRID_KEY"
firebase functions:config:set email.from="noreply@junkshop.com"
```

Running in Development Mode

Quick Start

```
# Start everything
npm run dev  # If script configured

# Or manually
firebase emulators:start
```

Access Points

- **Main Site**: http://localhost:5000
- **Admin Panel**: http://localhost:5000/admin.html
- **Payment Success**: http://localhost:5000/payment-success.html
- **Emulator UI**: http://localhost:4000

Using Test Data

1. **Populate Test Products**:
   - Login to admin panel (local)
   - Add sample products
   - Upload test images

2. **Test Orders**:
   - Use local checkout flow

   - Use SumUp test credentials
   - Verify webhook handling

Production Checklist

Before going live with real payments:

- [ ] Switch to **production** SumUp API key (not test)
- [ ] Remove test/debug console.logs from code
- [ ] Verify SSL certificate on custom domain (if using)
- [ ] Test with real card (small amount)
- [ ] Verify email confirmations work
- [ ] Set up monitoring and alerts
- [ ] Document admin procedures
- [ ] Train staff on admin panel
- [ ] Set up customer support email
- [ ] Add terms & conditions
- [ ] Add privacy policy
- [ ] Add refund policy
- [ ] Verify GDPR compliance
- [ ] Set up analytics tracking
- [ ] Configure domain (if using custom domain)

Additional Resources

Documentation

- **Firebase**: https://firebase.google.com/docs
- **SumUp API**: https://developer.sumup.com
- **Alpine.js**: https://alpinejs.dev
- **Tailwind CSS**: https://tailwindcss.com

Support

- **JunkShop Support**: junkshopdumfries@gmail.com
- **Firebase Support**: https://firebase.google.com/support
- **SumUp Support**: https://help.sumup.com

Monitoring

```
# View function logs
firebase functions:log

# View specific function
firebase functions:log --only createSumupCheckout

# View recent errors
firebase functions:log --only handleSumupWebhook | grep ERROR
```

Next Steps

After successful deployment:

1. **Test thoroughly** with SumUp test mode
2. **Switch to production** SumUp credentials
3. **Process test payment** with real card (refund immediately)
4. **Monitor logs** for first few days
5. **Gather customer feedback**
6. **Implement email notifications** (Phase 2)
7. **Add customer accounts** (Phase 3)
8. **Expand product catalog**

Status: ✅ Ready for Production Deployment
Last Tested: October 17, 2025
Deployment Target: https://junkshop-website-gem.web.app

For any issues during deployment, refer to the COMPREHENSIVE_BUG_ANALYSIS.md and check the troubleshooting section above.