

# JunkShop Website - Bug Report & Analysis

**Date:** October 17, 2025  
**Project:** JunkShop E-commerce Website  
**Technologies:** Firebase, Alpine.js, TailwindCSS, SumUp Payment Gateway

## Executive Summary

The JunkShop website is a well-designed e-commerce platform for antiques and collectibles. However, there are **critical bugs** related to payment processing and several issues that need to be addressed before the site can go live.

### Critical Issues (Must Fix Before Launch)

- 1. **Payment processing is completely non-functional** - Orders are saved without charging customers
- 2. **SumUp integration is incomplete** - Frontend doesn't connect to the payment gateway
- 3. **JavaScript syntax error** - Truncated function in index.html

### Priority Fixes Needed

- 1. Complete SumUp payment integration
- 2. Fix JavaScript errors
- 3. Implement proper error handling
- 4. Add order confirmation emails

## Detailed Bug Analysis

### CRITICAL BUG #1: No Payment Processing

**Location:** index.html - handleCheckout() function (lines ~810-820)

**Issue:**

The checkout process shows a fake loading screen but **NEVER processes any payment**. It directly saves the order to Firestore and marks products as sold without charging the customer.

**Current Code:**

```
handleCheckout() {
  this.isLoadingPayment = true;
  setTimeout(() => {
    this.placeOrder(); // <-- Goes directly to order placement
    this.isLoadingPayment = false;
  }, 2500);
}
```

**Impact: SEVERE** - Customers can place orders without paying, leading to:  
- Lost revenue

- Fraudulent orders
- Inventory marked as sold without payment
- Legal/financial issues

**Status:** ❌ NOT FIXED

---

## 🔴 CRITICAL BUG #2: SumUp Integration Missing

**Location:** Frontend ( `index.html` ) lacks Cloud Functions integration

**Issue:**

- Cloud Function `createSumupCheckout` exists in `functions/index.js`
- BUT frontend code doesn't import Firebase Functions SDK
- No code calls the `createSumupCheckout` function
- No redirect to SumUp checkout page

**What's Missing:**

1. Firebase Functions import in frontend
2. Call to `createSumupCheckout` Cloud Function
3. Redirect logic to SumUp payment page
4. Payment verification/callback handling

**Impact:** **SEVERE** - Payment gateway completely non-functional

**Status:** ❌ NOT FIXED

---

## 🔴 CRITICAL BUG #3: JavaScript Syntax Error

**Location:** `index.html` - line ~780

**Issue:**

The code has a truncated line ending with just `o` :

```
updateQuantity(productId, quantity) {
  const item = this.cartItems.find(item => item.id === productId);
  if (item) { if (quantity <= 0) { this.removeFromCart(productId); } else { item.quantity = quantity; } }
},
o
```

The `o` appears to be the start of `openProductModal` but was cut off.

**Impact:** **HIGH** - JavaScript parsing errors, potential crashes

**Status:** ❌ NOT FIXED

---

## 🟡 HIGH PRIORITY #4: Missing Firebase Functions Configuration

**Location:** `functions/index.js` - line 7

**Issue:**

```
const SUMUP_SECRET_KEY = functions.config().sumup.secret_key;
```

This expects the SumUp secret key to be set in Firebase environment config, but there's no documentation or setup script for this.

**Required Action:**

```
firebase functions:config:set sumup.secret_key="YOUR_SUMUP_SECRET_KEY"
firebase deploy --only functions
```

**Impact: HIGH** - Cloud Function will fail without proper configuration

**Status:** ⚠️ NEEDS CONFIGURATION

---

## 🟡 HIGH PRIORITY #5: No Order Confirmation Emails

**Issue:** After order placement, customers receive no confirmation email.

**Impact: MEDIUM-HIGH** - Poor user experience, customer support issues

**Recommendation:** Implement email notifications using:

- Firebase Extensions (Trigger Email)
- SendGrid/Mailgun integration
- Or native Firebase Functions with nodemailer

**Status:** ❌ NOT IMPLEMENTED

---

## 🟡 MEDIUM PRIORITY #6: No Payment Verification

**Location:** Order flow lacks payment status checks

**Issue:** Even if SumUp integration is added, there's no webhook or callback to verify payment success before marking orders as complete.

**Required:**

1. SumUp webhook endpoint
2. Payment status verification
3. Order confirmation only after successful payment
4. Handle failed payments gracefully

**Status:** ❌ NOT IMPLEMENTED

---

## 🟢 MINOR ISSUE #7: Firebase Security Rules Not Reviewed

**Location:** Firestore security rules (not in repository)

**Issue:** The code has error handling for security rule failures, suggesting rules might be restrictive.

**Recommendation:** Review and document Firestore security rules to ensure:

- Admin users can read/write all data
- Anonymous users can only read active products
- Orders are write-only for authenticated users

**Status:** ⚠️ NEEDS REVIEW

---

## ● MINOR ISSUE #8: Missing Environment Variables Documentation

**Issue:** No `.env.example` or configuration documentation for:

- Firebase project ID
- SumUp API keys
- Admin user credentials

**Recommendation:** Create configuration documentation

**Status:** ❌ NOT DOCUMENTED

---

## ● MINOR ISSUE #9: No Input Validation

**Location:** Checkout form ( `checkoutForm` )

**Issue:** Limited client-side validation for:

- Email format
- Postcode validation
- Phone number (not collected)
- XSS protection

**Recommendation:** Add proper validation using Alpine.js or a validation library

**Status:** ⚠️ NEEDS IMPROVEMENT

---

## ● MINOR ISSUE #10: No Loading States for Product Fetch

**Issue:** Products load silently; no skeleton loaders or progress indicators

**Impact:** **LOW** - Minor UX issue

**Recommendation:** Add loading skeletons while fetching products

**Status:** ⚠️ ENHANCEMENT NEEDED

---

# SumUp Integration - Complete Implementation Plan

---

## What Needs to Be Done:

### 1. Configure Firebase Functions

```
# Set SumUp API credentials
firebase functions:config:set sumup.secret_key="YOUR_SECRET_KEY"

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

### 2. Add Firebase Functions SDK to Frontend

**Update** `index.html` - Add to imports section:

```
import { getFunctions, httpsCallable } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-functions.js";
```

### 3. Initialize Functions in Frontend

```
window.firebaseShopServices = {
  db: null,
  auth: null,
  functions: null, // ADD THIS
  init: function() {
    const app = initializeApp(firebaseConfig);
    this.db = getFirestore(app);
    this.auth = getAuth(app);
    this.functions = getFunctions(app, 'europe-west2'); // ADD THIS
    // ... rest of init
  }
}
```

#### 4. Replace `handleCheckout()` Function

```

async handleCheckout() {
  if (this.cartItems.length === 0) {
    alert('Your cart is empty');
    return;
  }

  this.isLoadingPayment = true;

  try {
    // Calculate total in pence (SumUp requires smallest currency unit)
    const totalInPence = Math.round(this.cartTotal * 100);

    // Call Cloud Function to create SumUp checkout
    const { functions } = window.firebaseShopServices;
    const createCheckout = httpsCallable(functions, 'createSumupCheckout');

    const result = await createCheckout({
      amount: totalInPence,
      currency: 'GBP',
      description: `JunkShop Order - ${this.cartItems.length} items`,
      customerEmail: this.checkoutForm.email
    });

    // Save order as "pending" (will be updated after payment)
    await this.savePendingOrder(result.data.checkoutId);

    // Redirect to SumUp payment page
    window.location.href = `https://api.sumup.com/v0.1/checkouts/${result.data.checkoutId}`;

  } catch (error) {
    console.error('Checkout error:', error);
    alert(`Payment setup failed: ${error.message}`);
    this.isLoadingPayment = false;
  }
}

```

## 5. Add Pending Order Function

```

async savePendingOrder(checkoutId) {
  const { db, collection, doc, writeBatch, serverTimestamp } = window.firebase-
    ShopServices;

  const batch = writeBatch(db);

  const orderRef = doc(collection(db, "orders"));
  batch.set(orderRef, {
    customerDetails: this.checkoutForm,
    deliveryMethod: this.deliveryMethod,
    paymentMethod: 'Card (SumUp)',
    sumupCheckoutId: checkoutId,
    items: this.cartItems.map(item => ({
      id: item.id,
      name: item.name,
      quantity: item.quantity,
      price: item.price,
      postage: item.postage
    })),
    total: this.cartTotal,
    createdAt: serverTimestamp(),
    status: 'pending_payment' // Will be updated by webhook
  });

  await batch.commit();
}

```

## 6. Implement SumUp Webhook (Cloud Functions)

Add to `functions/index.js` :

```

exports.handleSumupWebhook = functions.region('europe-west2').https.onRequest(async
(req, res) => {
  if (req.method !== 'POST') {
    return res.status(405).send('Method Not Allowed');
  }

  const { id, status, checkout_reference } = req.body;

  if (status === 'PAID') {
    const admin = require('firebase-admin');
    const db = admin.firestore();

    // Find order by checkout ID
    const ordersSnapshot = await db.collection('orders')
      .where('sumupCheckoutId', '=', id)
      .limit(1)
      .get();

    if (!ordersSnapshot.empty) {
      const orderDoc = ordersSnapshot.docs[0];
      const orderData = orderDoc.data();

      // Update order status and mark products as sold
      const batch = db.batch();

      batch.update(orderDoc.ref, {
        status: 'paid',
        paidAt: admin.firestore.FieldValue.serverTimestamp()
      });

      // Mark products as sold
      orderData.items.forEach(item => {
        const productRef = db.collection('products').doc(item.id);
        batch.update(productRef, { status: 'sold' });
      });

      await batch.commit();

      // TODO: Send confirmation email here
    }
  }

  res.status(200).send('OK');
});

```

---

## Testing Checklist

---

Before going live, test:

- [ ] Firebase Functions deployment successful
- [ ] SumUp API credentials configured
- [ ] Checkout creates SumUp session
- [ ] Redirect to SumUp works
- [ ] Payment success updates order status
- [ ] Products marked as sold after payment
- [ ] Payment failure handled gracefully



- [ ] Order confirmation emails sent
  - [ ] Admin dashboard shows correct order status
  - [ ] Firestore security rules tested
- 

## Deployment Steps

---

### 1. Configure SumUp:

```
bash
firebase functions:config:set sumup.secret_key="YOUR_KEY"
```

### 2. Deploy Functions:

```
bash
firebase deploy --only functions
```

### 3. Deploy Frontend:

```
bash
firebase deploy --only hosting
```

### 4. Configure Webhook:

- In SumUp dashboard, set webhook URL to:









```
https://europe-west2-junkshop-website-gem.cloudfunctions.net/handleSumupWebhook
```

### 5. Test Payment Flow:

- Use SumUp test credentials
  - Complete test purchase
  - Verify order status updates
- 

## Recommended Next Steps

---

1.  **FIX CRITICAL BUGS** (Payment processing)
  2.  **Complete SumUp integration** (Following plan above)
  3.  **Add email notifications**
  4.  **Implement webhook verification**
  5.  **Add comprehensive error handling**
  6.  **Security audit** (Firestore rules, input validation)
  7.  **Performance testing**
  8.  **Create backup/restore procedures**
- 

## Additional Recommendations

---

### Security

- Implement rate limiting on order creation
- Add CAPTCHA to prevent bot orders
- Validate all user inputs server-side
- Use Firebase App Check

## User Experience

- Add order tracking page
- Implement customer account system
- Add wishlist functionality
- Improve mobile responsiveness

## Business

- Add analytics (Google Analytics 4)
- Implement inventory management
- Add discount codes/promotions
- Create admin reports dashboard

---

**Report prepared by:** DeepAgent

**Status:** Ready for fixes implementation