**# Nike Project: Comprehensive Testing & Debugging Guide**

## 1. Overview

This document provides a structured approach to testing and debugging the Nike project, focusing on integration points, API configurations, and deployment strategies.

## 2. Environment Configuration

### 2.1 API Key Management

- Centralize API keys in secure environment variables

- Use separate keys for development, staging, and production

- Implement key rotation mechanism

### 2.2 Required Environment Variables

| Service | Variable Name | Required | Purpose |

|--------------|--------------------------|----------|----------------------------|

| ShipEngine | `SHIPENGINE\_API\_KEY` | Yes | Shipping API Authentication |

| Stripe | `STRIPE\_SECRET\_KEY` | Yes | Payment Processing |

| Stripe | `STRIPE\_PUBLISHABLE\_KEY`| Yes | Client-side Payment |

## 3. API Integration Testing

### 3.1 ShipEngine API Testing

\*\*Objectives:\*\*

- Validate API connectivity

- Verify carrier and shipping calculations

- Test error handling

\*\*Test Scenarios:\*\*

1. Successful carrier retrieval

2. Shipping rate calculation

3. Label generation

4. Error response handling

### 3.2 Stripe Payment Integration

\*\*Objectives:\*\*

- Validate payment processing

- Test different payment scenarios

- Verify transaction security

\*\*Test Scenarios:\*\*

1. Successful payment creation

2. Payment method validation

3. Refund processing

4. Transaction error handling

## 4. Deployment Checklist

### 4.1 Pre-Deployment Validation

- [ ] All environment variables configured

- [ ] Local build successful

- [ ] API routes tested

- [ ] Error logging implemented

### 4.2 Vercel Deployment Considerations

- Use environment-specific configurations

- Enable detailed build logs

- Implement comprehensive error tracking

## 5. Debugging Strategies

### 5.1 Logging Approach

```javascript

// Recommended logging pattern

try {

// API call or critical operation

console.log('Operation Details:', {

service: 'ShipEngine',

action: 'Rate Calculation',

payload: sanitizedPayload

});

} catch (error) {

console.error('Detailed Error:', {

service: error.service,

code: error.code,

message: error.message

});

}

```

## 6. Common Error Resolution

### 6.1 ShipEngine Integration Errors

\*\*Error:\*\* API Key Validation Failure

\*\*Solutions:\*\*

- Verify API key format

- Check API key permissions

- Confirm environment variable configuration

### 6.2 Stripe Processing Errors

\*\*Error:\*\* Payment Authentication Failure

\*\*Solutions:\*\*

- Validate Stripe configuration

- Check API key permissions

- Implement robust error handling

## 7. Monitoring & Performance

### 7.1 Recommended Tools

- Vercel Performance Monitoring

- Stripe Webhook Validation

- ShipEngine API Logs

## 8. Security Considerations

- Use environment-specific API keys

- Implement rate limiting

- Sanitize all user inputs

- Use HTTPS for all API communications

## 9. Continuous Improvement

- Regularly update dependencies

- Conduct periodic security audits

- Maintain comprehensive test coverage

---

\*\*Prepared By:\*\* Project Development Team

\*\*Version:\*\* 1.0

\*\*Last Updated:\*\* [Current Date]