

SafeSpace - Comprehensive Testing Document

Pre-Testing Setup

1. Database Reset

bash

Stop existing containers

docker-compose down -v

Restart fresh

docker-compose up -d

Wait for database to be ready

sleep 10

2. Create Test Accounts

- **Moderator:** username: admin, password: admin123 (default)
 - **Users:** Create 5 regular users (Alice, Bob, Charlie, Diana, Eve)
-

Testing Checklist

Phase 1: Authentication & Registration (15 minutes)

Test 1.1: User Registration

- Register new user "Alice" with valid email
- Verify welcome email received
- Check email contains: username, login URL, SafeSpace branding
- Try registering with same username (should fail)
- Try registering with same email (should fail)
- Register as moderator "ModTest"

Test 1.2: User Login

- Login as Alice
- Verify redirect to /feed
- Login as admin (moderator)
- Verify redirect to /mod-dashboard
- Test wrong password (should fail)
- Test non-existent user (should fail)

Expected Results:

-  Welcome emails sent

- ☒ Proper role-based redirects
 - ☒ Duplicate prevention works
-

Phase 2: Abuse Detection System (30 minutes)

Test 2.1: Clean Comments (Should Auto-Approve)

Login as Alice, post comments on Bob's posts:

1. "This is a great post! Thanks for sharing."
2. "I completely agree with your perspective."
3. "Could you explain this further? Very interesting!"

Expected: All comments visible immediately (blue boxes for moderator)

Test 2.2: Positive Context with Flagged Words (Should Auto-Approve)

1. "This tutorial is fucking brilliant! Saved me hours."
2. "That's stupid simple to understand, love it!"
3. "Damn good explanation, thanks!"

Expected: Auto-approved despite containing flagged words

Test 2.3: Clearly Abusive Comments (Should Auto-Hide)

1. "You are an asshole"
2. "You fucking idiot"
3. "Shut up bitch"
4. "Go to hell"
5. "You stupid moron"

Expected:

- ☒ Comments hidden immediately (red boxes)
- ☒ NOT visible in public feed
- ☒ Confidence score > 90%

Test 2.4: Character Substitution Detection

1. "You st*pid idiot"
2. "F**k you"
3. "You @sshole"
4. "S**t post"
5. "Fuc|< off"

Expected: All detected and hidden

Test 2.5: Repeated Characters Detection

1. "Stuuuuupid post"

2. "Idiiiiiot"
3. "Fuuuuck this"

Expected: All detected and hidden

Test 2.6: Case Insensitivity Testing





1. "GO TO HELL"
2. "Go To Hell"
3. "go to hell"
4. "gO tO hELL"

Expected: ALL variations should be detected (test critically!)

Test 2.7: Uncertain Comments (Should Need Human Review)

1. "This stupid process is confusing"
2. "What the hell is this about?"
3. "This is a bit dumb but..."

Expected:

-  Yellow boxes in moderator view
-  Status: "NEEDS REVIEW"
-  NOT visible in public feed
-  Appears in Review Comments queue

Test 2.8: Multiple Flagged Words (Auto-Hide)

1. "You stupid fucking idiot bitch"

Expected: Auto-hidden (3+ flagged words rule)

Phase 3: Spam Detection System (30 minutes)

Test 3.1: Non-Promotional Repetition

Login as Charlie, comment on same post:

Post 1: "Great content!"


Post 2: "Great content!"

Post 3: "Great content!"

Post 4: "Great content!"

Post 5: "Great content!"

Post 6: "Great content!"  Warning

Post 7: "Great content!"  SPAM (hide last 2)

Expected:

- Comments 1-5: Approved

- Comment 6: Approved with warning
- Comment 7: Hidden as spam

Test 3.2: Promotional Content Spam

Login as Diana, comment on same post:

Post 1: "Check out my website example.com"

Post 2: "Check out my website example.com"

Post 3: "Check out my website example.com"

Post 4: "Check out my website example.com" ⚠ Warning

Post 5: "Check out my website example.com" 🚫 ALL 5 HIDDEN

Expected:

- Comments 1-3: Approved
- Comment 4: Approved with warning
- Comment 5: All 5 promotional comments hidden

Test 3.3: URL Detection

1. "Visit https://spam.com for deals"

2. "Check www.promo.net"

3. "Go to example.com"

Expected: All flagged as promotional

Test 3.4: Promotional Keywords

1. "DM me for more info"

2. "Follow me on Instagram"

3. "Buy now, limited offer!"

4. "Work from home opportunity"

Expected: All flagged as promotional

Phase 4: User Warning & Suspension System (20 minutes)

Test 4.1: Abuse Rate Monitoring

Using Alice's account, create pattern:

- Post 10 clean comments
- Post 10 abusive comments
- **Total: 50% abuse rate**


Check:

- Navigate to Moderator → Statistics → Select Alice
- Verify abuse rate shows ~50%

- Check Alice's email for warning email

Test 4.2: Warning Email

Expected Warning Email Contents:





-  Warning icon
- Current abuse rate percentage
- Threshold information (70%)
- Community guidelines reminder
- Tips for improvement

Test 4.3: Trigger Suspension (70% Abuse Rate)

Continue with Alice:


- Post 5 more abusive comments
- **New total: ~70% abuse rate**

Expected:

-  Account suspended automatically
-  Suspension email sent
-  Cannot login (show error message)
-  Suspension reason visible in moderator dashboard

Test 4.4: Suspension Email

Expected Suspension Email Contents:

-  Suspension notification
- Reason for suspension
- Appeal process instructions
- Appeal email address
- Cannot login message

Phase 5: Moderator Dashboard (30 minutes)

Test 5.1: User Management

- View all users list
- Check user statistics (posts, comments, flagged)
- Filter by specific user
- Verify abuse rate calculations
- Check last activity tracking

Test 5.2: All Posts View

- Select user from dropdown

- View all their posts
- Check comment counts (approved/pending/hidden)
- Click "View Comments" on a post
- Verify color coding: Blue (approved), Yellow (pending), Red (hidden)

Test 5.3: Review Comments Queue

- Check posts with pending comments
- Verify user abuse rate displayed
- View comments needing review
- Test Approve action
- Test Hide action
- Test Delete action
- Verify changes reflect immediately

Test 5.4: Flagged Comments

- Filter flagged comments by user
- Verify only hidden/flagged comments shown
- Check flagged words displayed
- Verify confidence scores

Test 5.5: Statistics Dashboard

System Statistics:

- Total users, posts, comments
- Clean comments count
- Flagged comments count
- Spam comments count ★ NEW
- AI efficiency percentage
- Abuse detection rate
- Spam detection rate ★ NEW

User-Specific Statistics:

- Select user from dropdown
- Verify user profile information
- Check post/comment counts
- Verify abuse rate circle visualization
- Verify spam rate circle visualization ★ NEW
- Check rate descriptions (low/medium/high)

Test 5.6: Account Deletion ★ NEW FEATURE

- Navigate to User Management
- Click "Delete Account" on a user
- Verify deletion impact preview:
 - Posts to delete
 - Comments to delete
 - Comments on posts to delete
 - Total affected
 - Blocks to remove
- Type username to confirm
- Confirm deletion
- Verify deletion confirmation email sent
- Verify user removed from list
- Try to login as deleted user (should fail)
- Verify all user data removed from database

Test Edge Cases:

- Try deleting own account (should fail)
- Try deleting another moderator (should fail)




Phase 6: User Feed Experience (20 minutes)


Test 6.1: My Posts View

Login as regular user:

- Create new post (max 2000 chars)
- Verify character counter
- View created post
- See only approved comments on own posts
- Verify comment count accurate

Test 6.2: Explore Feed


- View posts from other users
- See only approved comments
- Comment on post
- Verify comment submission feedback:
 -  Clean comment → Success message
 -  Warning → Warning message shown
 -  Spam → Spam detection message

-  Pending → Review message

Test 6.3: Comment Submission Responses


Test these scenarios and verify correct messages:

Clean Comment:

Message: "  Comment posted successfully!"


Visible: Yes, immediately

Spam Detected:

Message: "  Your comment has been detected as spam and hidden."


Visible: No

Spam Warning:

Message: "  Warning: You've posted similar content X times..."


Visible: Yes

Abuse Detected:

Message: "  Your comment has been flagged as potentially abusive..."

Visible: No

Needs Review:

Message: "  Your comment has been submitted for moderation review."

Visible: No (until approved)

Phase 7: Email System Testing (25 minutes)

Test 7.1: Welcome Email

- Register new user
- Check email inbox
- Verify email received within 30 seconds
- Check email formatting
- Test "Start Exploring" link
- Verify sender: SafeSpace Moderation System

Test 7.2: Warning Email

- Trigger warning condition (50%+ abuse rate)
- Check user's email
- Verify warning email contents:
 - Current abuse rate
 - Threshold information
 - Guidelines reminder

- Improvement tips

Test 7.3: Suspension Email

- Trigger suspension (70%+ abuse rate)
- Check user's email
- Verify suspension email contents:
 - Suspension reason
 - Appeal process
 - Contact information

Test 7.4: Account Deletion Email 🌟 NEW

- Delete a user account as moderator
- Check deleted user's email
- Verify deletion confirmation:
 - Account deletion notice
 - Data removal confirmation
 - Support contact info

Test 7.5: Blocking Notification (If Implemented)

- Create scenario where user gets blocked
 - Check blocked user's email
 - Verify blocking notification contents
-

Phase 8: Edge Cases & Error Handling (20 minutes)

Test 8.1: Input Validation

- Empty post/comment (should fail)
- Post > 2000 characters (should fail)
- Comment > 1000 characters (should fail)
- Special characters in username
- SQL injection attempts (should be safe)

Test 8.2: Authentication Edge Cases

- Expired token (logout after token expiry)
- Invalid token
- Access moderator routes as user (should fail)
- Access after logout

Test 8.3: Concurrent Operations

- Multiple users commenting simultaneously

- Moderator approving while user viewing
- Delete post while someone is commenting

Test 8.4: Database Consistency

- Delete user → verify orphaned data removed
 - Delete post → verify comments removed
 - User suspension → verify posts still visible
-

Phase 9: Performance & Scalability (15 minutes)

Test 9.1: Load Testing

Create test data:

python

Quick script to generate test data

import requests

BASE_URL = "http://localhost:8001"

Create 20 users

for i in range(20):

```
requests.post(f"{BASE_URL}/api/register", json={
    "username": f"testuser{i}",
    "email": f"test{i}@example.com",
    "password": "test123",
    "role": "user"
})
```

Each user creates 5 posts

Each post gets 10 comments

Check:

- Dashboard loads quickly (<2 seconds)
- Statistics calculated correctly
- Filtering works smoothly
- No database errors in logs

Test 9.2: Large Comment Load

- Create post with 50+ comments

- Open in user view
 - Open in moderator view
 - Verify scrolling works
 - Check performance
-

Phase 10: UI/UX Testing (10 minutes)

Test 10.1: Responsive Design

Test on different screen sizes:

- Desktop (1920x1080)
- Tablet (768px)
- Mobile (375px)

Test 10.2: Animations & Transitions

- Tab switching smooth
- Modal animations work
- Button hover effects
- Loading spinners display

Test 10.3: Accessibility

- Tab navigation works
 - Focus indicators visible
 - Color contrast adequate
 - Error messages clear
-

Critical Test Cases Summary

✅ Must Pass Before Presentation

1. Abuse Detection:

- "Go to hell" → Auto-hide ✅
- "GO TO HELL" → Auto-hide ✅
- "go to HELL" → Auto-hide ✅

2. Spam Detection:

- 7th repetition → Hide
- 5th promotional → Hide ALL 5

3. Email System:

- Welcome email → Within 30 seconds
- Warning email → At 50% abuse rate

- Suspension email → At 70% abuse rate
- Deletion email → On account deletion

4. **User Suspension:**

- Cannot login when suspended
- Error message shows reason

5. **Moderator Actions:**

- Approve → Comment becomes visible
- Hide → Comment becomes hidden
- Delete → Comment removed permanently
- Delete account → All user data removed

6. **Statistics:**

- All percentages calculated correctly
- Abuse rate matches actual data
- Spam rate matches actual data

Test Data Generation Script

Use this to create clean, demonstration-ready data:

```
python
```

```
# generate_test_data.py
```

```
import requests
```

```
import random
```

```
import time
```

```
BASE_URL = "http://localhost:8001"
```

```
# Sample clean comments
```

```
clean_comments = [
```

```
    "Great post! Very informative.",
```

```
    "Thanks for sharing this perspective.",
```

```
    "I completely agree with your points.",
```

```
    "This is really helpful, appreciate it!",
```

```
    "Excellent explanation, well done.",
```

```
]
```

Sample posts

```
sample_posts = [  
    "Just finished reading an amazing book on AI ethics. Highly recommend!",  
    "Does anyone have tips for improving productivity while working from home?",  
    "Sharing my experience with learning Vue.js - it's been a great journey!",  
    "Looking for recommendations on project management tools. What do you use?",  
    "Just deployed my first full-stack application. Feeling accomplished!",  
]
```

def create_demo_data():

"""Generate clean, presentation-ready test data"""

Create 5 regular users

users = []

for i in range(1, 6):

username = ["Alice", "Bob", "Charlie", "Diana", "Eve"][i-1]

try:

response = requests.post(f"{BASE_URL}/api/register", json={

"username": username,

"email": f"{username.lower()}@example.com",

"password": "demo123",

"role": "user"

})

if response.status_code == 200:

users.append(username)

print(f"✅ Created user: {username}")

except Exception as e:

print(f"❌ Failed to create {username}: {e}")

time.sleep(1)

Each user creates 2-3 posts

for username in users:

Login

login_response = requests.post(f"{BASE_URL}/api/login", json={

```
"username": username,  
"password": "demo123"  
})
```

```
if login_response.status_code == 200:  
    token = login_response.json()["access_token"]  
    headers = {"Authorization": f"Bearer {token}"}
```

```
# Create posts
```

```
num_posts = random.randint(2, 3)  
for _ in range(num_posts):  
    post_content = random.choice(sample_posts)  
    requests.post(f"{BASE_URL}/api/posts/",  
                  json={"content": post_content},  
                  headers=headers)  
    time.sleep(0.5)
```

```
print(f"✅ Created {num_posts} posts for {username}")
```

```
# Add clean comments from each user to other users' posts
```

```
print("\n 📝 Adding clean comments...")
```

```
# Implementation here...
```

```
print("\n✅ Demo data generation complete!")
```

```
print(f"Created: {len(users)} users, ~{len(users) * 2.5} posts")
```

```
if __name__ == "__main__":  
    create_demo_data()
```

Testing Execution Timeline (3 hours total)

Day 1 - Setup & Core Testing (2 hours)

- Phase 1: Authentication (15 min)
- Phase 2: Abuse Detection (30 min)
- Phase 3: Spam Detection (30 min)

- Phase 4: User Warning/Suspension (20 min)
- Phase 5: Moderator Dashboard (30 min)

Day 2 - Advanced Testing (1 hour)

- Phase 6: User Feed (20 min)
- Phase 7: Email System (25 min)
- Phase 8: Edge Cases (15 min)

Day 3 - Final Checks (Before Presentation)

- Generate fresh demo data
- Quick smoke test of all features
- Verify emails working

Bug Report Template

If issues found during testing:

markdown

Bug Report

****Bug ID:**** BUG-001

****Severity:**** High/Medium/Low

****Module:**** Abuse Detection / Spam Detection / etc.

****Description:****

Clear description of the issue

****Steps to Reproduce:****

1. Login as Alice
2. Post comment "X"
3. Observe behavior

****Expected Result:****

What should happen

****Actual Result:****

What actually happened

****Screenshots:****

[If applicable]

****Status:**** Open/Fixed/Testing

Pre-Presentation Checklist

48 Hours Before:

- Complete all testing phases
- Fix critical bugs
- Test email delivery (all types)
- Verify database clean

24 Hours Before:

- Generate fresh demo data
- Clear all test data
- Verify all features working
- Check email inbox empty

2 Hours Before:

- Restart containers
 - Quick smoke test
 - Have backup database ready
 - Test presentation flow
-

Known Issues to Document

If any issues persist, document them clearly:

Example:

Issue: Case sensitivity in "Go to hell"

Status: Tested - Working correctly

Note: All variations (GO TO HELL, go to hell, Go To Hell)

are properly detected and hidden