SecureMed Troubleshooting Guide

Healthcare Cybersecurity & HIPAA Compliance Platform

Version: 1.0 - Final

Release Date: December 2025

Project: SecureMed - Comprehensive Healthcare Security & HIPAA Compliance Management System

Institution: Florida International University, Knight Foundation School of Computing and Information Sciences

Course: CIS 4914 - Cybersecurity Capstone Project II

Table of Contents

1.0 Introduction

Welcome to the SecureMed Troubleshooting Guide. This document provides solutions to the most common issues encountered during installation, setup, and operation of the SecureMed platform.

How to Use This Guide

1. **Identify Your Problem**: Look for symptoms that match your issue
2. **Find the Solution**: Follow step-by-step instructions
3. **Verify Resolution**: Test that the fix works

4. **Preventive Measures**: Apply suggestions to avoid future issues

Documentation Scope

This guide covers issues identified during **Sprints 3-6** testing:

- Environment and dependency conflicts
- Application startup failures
- Database access problems
- Frontend/backend integration issues
- Functionality glitches
- Security scanning problems
- PDF generation failures
- Browser compatibility issues

When This Guide Helps

 Installation problems

 Flask server won't start

 Templates not found

 Port already in use

 Database errors

 Session timeout issues

 Training scores not updating

 Task assignments failing

 PDF reports not generating

When You Need Technical Support

 System crashes repeatedly

 Data appears corrupted

 Security vulnerability suspected

 Multiple issues simultaneously

See Section 13.0: When to Seek Help

2.0 Quick Reference Guide

2.1 The Most Common Issues (Top 5)

Issue	Solution	Time to Fix
Flask won't start (ModuleNotFoundError)	Activate venv, run `pip install -r requirements.txt`	2 minutes

Issue	Solution	Time to Fix
Port 5000 already in use	Kill process: `lsof -ti:5000 | xargs kill -9`	30 seconds
Template not found (login.html)	Verify correct directory: `cd /path/to/Cap_Finaldev`	1 minute
Database locked error	Restart Flask: `Ctrl+C` then `python webapp.py`	30 seconds
Session timeout too fast	Expected (2-min demo mode) - document for users	N/A

2.2 System Requirements Checklist

Before troubleshooting, verify:

- □ Python 3.8+ installed: `python --version`
- □ pip installed: `pip --version`
- □ Virtual environment support: `python -m venv --help`
- □ 200+ MB disk space available: `df -h`
- □ Port 5000 not in use: `lsof -i :5000` (macOS/Linux)
- □ Internet connection (for pip installations)

2.3 Emergency Reset (Nuclear Option)

If everything is broken, do a complete reset:

```
# Stop Flask if running
Ctrl + C

# Kill all Python processes
pkill -9 -f "webapp.py"         # macOS/Linux
taskkill /F /IM python.exe     # Windows (use Task Manager)

# Remove virtual environment
rm -rf venv                      # macOS/Linux
rmdir /s /q venv               # Windows

# Remove database
rm securemed.db                # macOS/Linux
del securemed.db              # Windows

# Start fresh
python3 -m venv venv
source venv/bin/activate      # macOS/Linux OR venv\Scripts\activate (Windows)
pip install -r requirements.txt
python webapp.py
```

This completely resets SecureMed to a clean state.

3.0 Environment & Dependency Issues

3.1 ModuleNotFoundError: No module named 'flask'

Symptoms

```
Traceback (most recent call last):
  File "webapp.py", line 1, in <module>
    from flask import Flask
ModuleNotFoundError: No module named 'flask'
```

Application fails immediately on startup

Causes

- Virtual environment not activated
- Dependencies not installed
- Wrong Python version

- pip cache corrupted

Solution (Step-by-Step)

Step 1: Verify Python Version

```
python --version
```

Output should be: `Python 3.8.x` or higher

If not: Install Python 3.8+ from python.org

Step 2: Activate Virtual Environment

On **macOS/Linux**:

```
source venv/bin/activate
```

On **Windows (PowerShell)**:

```
venv\Scripts\activate
```

On **Windows (Command Prompt)**:

```
venv\Scripts\activate.bat
```

Success indicator: Your prompt should show `(venv)` at the start:

```
(venv) user@computer Cap_Finaldev %
```

Step 3: Verify Virtual Environment Activation

```
# Should show path to venv
which python        # macOS/Linux
where python        # Windows

# Output should include: /path/to/Cap_Finaldev/venv/bin/python
```

Step 4: Install Dependencies

```
pip install -r requirements.txt
```

Expected output:

```
Collecting flask==3.1.2
Downloading flask-3.1.2-py3-none-any.whl
...
Successfully installed flask-3.1.2 cryptography-46.0.3 reportlab-4.4.4 ...
```

Step 5: Verify Installation

```
# Test Flask import
python -c "import flask; print(flask.__version__)"

# Output: 3.1.2
```

Step 6: Start Flask

```
python webapp.py
```

Expected output:

```
 * Running on http://127.0.0.1:5000
 * Press CTRL+C to quit
```

Prevention

- Always activate virtual environment FIRST before installing or running
- Re-run `pip install -r requirements.txt` after pulling code updates
- Don't use system Python - always use virtual environment

Related Issues

- "Permission denied" when installing: Use `pip install --user` or check directory permissions
- "Upgrade pip" warning: Run `pip install --upgrade pip` to fix
- Slow installation: Check internet connection

3.2 ModuleNotFoundError: No module named 'cryptography'

Symptoms

```
ModuleNotFoundError: No module named 'cryptography'
```

Error occurs when trying to encrypt/decrypt SSN

Causes

- cryptography library not installed
- Virtual environment not activated
- Installation failed silently

Solution

Quick Fix:

```
# Ensure venv is activated
source venv/bin/activate  # macOS/Linux

# Reinstall cryptography
pip install cryptography --force-reinstall

# Restart Flask
python webapp.py
```

If still failing:

```
# Check what's installed
pip list | grep cryptography

# If missing, install with upgrade
pip install --upgrade cryptography

# Test import
python -c "from cryptography.fernet import Fernet; print('OK')"
```

3.3 Virtual Environment Not Found

Symptoms

```
source: venv/bin/activate: No such file or directory
# OR
'venv\Scripts\activate' is not recognized
```

Cannot activate virtual environment

Causes

- venv directory not created

- Working in wrong directory
- Accidentally deleted venv folder

Solution

Step 1: Verify Directory

```
pwd                    # macOS/Linux
cd                     # Windows
```

Should show: `/path/to/Cap_Finaldev` or `C:\path\to\Cap_Finaldev`

If not correct:

```
cd /path/to/Cap_Finaldev
```

Step 2: List Directory Contents

```
ls -la              # macOS/Linux
dir                 # Windows
```

Look for: `venv` folder should exist

Step 3: Create Virtual Environment

If venv doesn't exist:

```
python3 -m venv venv
```

Step 4: Activate

```
source venv/bin/activate  # macOS/Linux
venv\Scripts\activate     # Windows
```

Step 5: Reinstall Dependencies

```
pip install -r requirements.txt
```

Prevention

- Never delete the `venv` folder
- Always check you're in the correct directory
- Use `pwd` (macOS/Linux) or `cd` (Windows) to verify location

3.4 Pip Install Fails / Requirements Not Installing

Symptoms

```
ERROR: Could not find a version that satisfies the requirement
pip: command not found
ERROR: Permission denied
```

Cannot install dependencies

Causes

- Incorrect Python/pip version
- Network connectivity issues
- Corrupted pip cache
- Permission/authorization issues

Solution

Option 1: Clear Pip Cache & Reinstall

```
pip cache purge
pip install -r requirements.txt --no-cache-dir
```

Option 2: Upgrade Pip

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

Option 3: Install with User Flag (if permission denied)

```
pip install --user -r requirements.txt
```

Option 4: Check Network

```
# Test connectivity
ping google.com

# Test pip repository
pip install --upgrade pip --index-url https://pypi.org/simple/
```

Option 5: Manually Install

If all else fails, install individually:

```
pip install flask==3.1.2
pip install cryptography
pip install reportlab==4.4.4
pip install requests
```

4.0 Application Startup Issues

4.1 TemplateNotFound: login.html

Symptoms

```
jinja2.exceptions.TemplateNotFoundError: login.html
```

Error appears in terminal when starting Flask

Browser shows 500 error

Causes

- Flask running from wrong directory
- Project folder structure incorrect
- `templates` directory missing or moved

Solution (Step-by-Step)

Step 1: Stop Flask

```
Ctrl + C
```

Step 2: Verify Directory Structure

Navigate to project root:

```
cd /path/to/Cap_Finaldev
```

List contents:

```
ls -la              # macOS/Linux
dir                 # Windows
```

You should see:

```
webapp.py
requirements.txt
templates/          ← This folder must exist
static/             ← This folder must exist
securemed.db
venv/
```

Step 3: Check Templates Folder

```
ls -la templates/    # macOS/Linux
dir templates        # Windows
```

Should contain:

```
login.html
dashboard_react.html
user_dashboard_react.html
edr.html
training_simulator.html
patients.html
audit_trail.html
```

If templates folder is missing:

Download template files from project repository or restore from backup.

Step 4: Verify webapp.py References

Check that `webapp.py` has correct template path at top:

```
app = Flask(__name__, template_folder='templates')
```

This tells Flask where to find templates.

Step 5: Restart Flask

```
python webapp.py
```

Expected output:

```
 * Running on http://127.0.0.1:5000
```

Step 6: Test

Open browser: `http://127.0.0.1:5000/login`

Should see: SecureMed login page

Prevention

- Never move or rename the `templates` folder
- Always run Flask from project root directory
- Verify directory structure before each session

Troubleshooting Checklist

- □ Correct directory: `pwd` shows `/path/to/Cap_Finaldev`
- □ Templates folder exists: `ls templates` shows files
- □ webapp.py in root: `ls webapp.py` returns file
- □ Virtual environment activated: Prompt shows `(venv)`

4.2 Port Already in Use (OSError: [Errno 48] Address already in use)

Symptoms

```
OSError: [Errno 48] Address already in use
# OR
OSError: [WinError 10048] Only one usage of each socket address
```

Flask fails to start

Cannot bind to port 5000

Causes

- Another Flask instance running
- Another application using port 5000
- Process not fully shut down from previous run
- Port held by OS (rare)

Solution (Choose One)

Option 1: Kill Previous Flask Process (Recommended)

On **macOS/Linux**:

```
# Find process using port 5000
lsof -ti:5000

# Kill the process
lsof -ti:5000 | xargs kill -9

# Verify it's gone
lsof -ti:5000  # Should return nothing
```

On **Windows (PowerShell)**:

```
# Find process
netstat -ano | findstr :5000

# Kill the process (replace <PID> with actual number)
taskkill /PID <PID> /F

# Example: taskkill /PID 12345 /F
```

On **Windows (Command Prompt)**:

```
netstat -ano | findstr :5000
taskkill /PID <PID> /F
```

Option 2: Use Different Port

If you want to keep other process running:

Edit `webapp.py`:

```
# Find this line near bottom:
if __name__ == '__main__':
    app.run(debug=True, port=5000)  # ← Change 5000 to 5001, 5002, etc.

# Change to:
if __name__ == '__main__':
    app.run(debug=True, port=5001)
```

Then run Flask and access at:

```
http://127.0.0.1:5001/login
```

Option 3: Restart Computer (Nuclear Option)

If you can't find the process:

```
# macOS
sudo killall -9 python

# Or restart computer entirely
```

Verification

After fixing, verify port is free:

```
# macOS/Linux
lsof -i :5000  # Should return nothing

# Windows
netstat -ano | findstr :5000  # Should return nothing
```

Then start Flask normally:

```
python webapp.py
```

Prevention

- Always use `Ctrl + C` to stop Flask (don't close terminal)
- Don't run multiple Flask instances
- Close previous terminal windows before opening new ones

4.3 RuntimeError: Click will abort further execution

Symptoms

```
RuntimeError: Click will abort further execution because Python
was configured to use UTF-8 mode by default.
```

Occurs on some Windows systems

Causes

- Python UTF-8 mode conflict
- Windows locale settings
- Flask-Click compatibility issue

Solution

Option 1: Add Environment Variable

On **Windows (PowerShell)**:

```
$env:PYTHONIOENCODING = "utf-8"
python webapp.py
```

On **Windows (Command Prompt)**:

```
set PYTHONIOENCODING=utf-8
python webapp.py
```

Option 2: Modify webapp.py

Add at the very top of `webapp.py`:

```
import os
os.environ['PYTHONIOENCODING'] = 'utf-8'

from flask import Flask
# ... rest of code
```

Option 3: Upgrade Flask

```
pip install --upgrade flask
```

4.4 SyntaxError in webapp.py

Symptoms

```
SyntaxError: invalid syntax
File "webapp.py", line XX
```

Python cannot parse the code

Causes

- File corrupted during download
- Code editor saved in wrong format
- Merge conflict markers left in file

Solution

Step 1: Check for Merge Conflicts

Open `webapp.py` in text editor and search for:

```
<<<<<<< HEAD
```

If found, the file has merge conflicts. Resolve manually or restore clean copy.

Step 2: Verify File Format

File should be UTF-8 plain text, not Word document or Rich Text.

Step 3: Restore Clean Copy

If syntax error, restore from git:

```
git checkout webapp.py
```

Or re-download from repository.

Step 4: Verify Python Syntax

Test syntax without running:

```
python -m py_compile webapp.py

# No output means syntax is OK
```

5.0 Database Issues

5.1 sqlite3.OperationalError: database is locked

Symptoms

```
sqlite3.OperationalError: database is locked
```

Cannot read or write to database

Application crashes when accessing database

Causes

- Multiple Flask instances accessing same database file
- Database file corrupted
- Long-running transaction not completed
- File permissions issue

Solution (Step-by-Step)

Step 1: Stop Flask

```
Ctrl + C
```

Step 2: Kill All Python Processes

On **macOS/Linux**:

```
pkill -9 -f "webapp.py"
pkill -9 -f "python"
```

On **Windows**:

```
taskkill /F /IM python.exe
```

Step 3: Remove Database Lock Files

Check for temporary lock files:

```
ls -la securemed.db*      # macOS/Linux
dir securemed.db*         # Windows
```

If you see `securemed.db-journal`, delete it:

```
rm securemed.db-journal    # macOS/Linux
del securemed.db-journal   # Windows
```

Step 4: Restart Flask

```
python webapp.py
```

Step 5: Test Database Access

Open browser and try logging in:

```
http://127.0.0.1:5000/login
```

Prevention

- Never run multiple Flask instances simultaneously
- Always stop Flask with `Ctrl + C` (don't force-kill)
- Don't access database with other tools while Flask is running

If Still Failing

Option 1: Backup and Reset Database

```
# Backup current database
cp securemed.db securemed.db.backup

# Delete corrupted database
rm securemed.db

# Restart Flask (will auto-create new database)
python webapp.py

# Re-run Quick Setup to populate demo data
```

Option 2: Check File Permissions

```
# macOS/Linux
ls -la securemed.db

# Should show: -rw-r--r-- (user readable and writable)

# If not, fix permissions:
chmod 644 securemed.db
```

5.2 sqlite3.DatabaseError: file is not a database

Symptoms

```
sqlite3.DatabaseError: file is not a database
```

Database appears corrupted

Causes

- Database file deleted or replaced
- Disk full during database operations
- Unauthorized file modification

Solution

Option 1: Backup and Reset (Recommended)

```
# Backup
mv securemed.db securemed.db.corrupt

# Restart Flask (creates new database)
python webapp.py
```

Option 2: Repair Using SQLite Command Line

```
# Check database integrity
sqlite3 securemed.db "PRAGMA integrity_check;"

# If output shows errors, database is corrupted
# Only option is to restore from backup or recreate
```

Option 3: Restore from Backup

If you have a backup:

```
cp securemed.db.backup securemed.db
python webapp.py
```

Prevention

- Regularly backup `securemed.db`
- Monitor disk space (ensure >100 MB free)
- Don't modify database file manually

5.3 Table Creation Failed on Startup

Symptoms

```
Error creating tables
OperationalError: [existing table]
```

Database exists but tables missing

Causes

- Database partially initialized
- Startup script ran but failed
- Corrupted database from previous run

Solution

Option 1: Delete Database and Restart

```
# Stop Flask
Ctrl + C

# Delete corrupted database
rm securemed.db

# Restart Flask (auto-creates fresh database)
python webapp.py
```

Option 2: Manual Table Initialization

If you want to fix without losing data:

```
# In Python console (advanced)
import sqlite3
conn = sqlite3.connect('securemed.db')
cursor = conn.cursor()

# Check existing tables
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
print(cursor.fetchall())

# If tables missing, re-run setup_database.py
```

6.0 Frontend/Backend Integration Issues

6.1 React Components Not Rendering

Symptoms

- Blank white page
- Dashboard loads but no content
- Components missing or showing errors in console

Causes

- React not loaded from CDN
- JavaScript error in browser
- Flask not serving HTML properly

Solution

Step 1: Open Browser Console

- **Chrome/Firefox**: Press `F12`
- **Safari**: Menu → Develop → Show Web Inspector
- Look for red error messages

Step 2: Check Common Errors

Error: "React is not defined"

- React CDN link missing in HTML
- Solution: Verify `<script src="https://cdn.jsdelivr.net/npm/react...">` tag in template

Error: "Cannot read property 'render' of null"

- Element with ID not found in HTML
- Solution: Check HTML has `<div id="root"></div>` or similar

Error in Console (CORS error)

```
Access to XMLHttpRequest at 'http://127.0.0.1:5000/api/...'
from origin 'http://127.0.0.1:5000' has been blocked by CORS policy
```

- Solution: Flask-CORS needs configuration
- Add to `webapp.py`: `CORS(app)`

Step 3: Hard Refresh Browser

Clear cache and reload:

- **Chrome/Windows**: `Ctrl + Shift + Delete`
- **Chrome/macOS**: `Cmd + Shift + Delete`

- Then `Ctrl + F5` (hard refresh) or `Cmd + Shift + R`

Step 4: Check Network Tab

In DevTools, click "Network" tab:

- Look for failed requests (red)
- Click failed request to see error details
- Common issues: 404 (file not found), 500 (server error)

Step 5: Restart Flask and Browser

```
# Stop Flask
Ctrl + C

# Restart Flask
python webapp.py

# Close all browser tabs
# Open new tab: http://127.0.0.1:5000/login
```

Prevention

- Keep all CDN links current
- Test after code changes
- Monitor console for errors during development

6.2 Buttons Not Responding (Assignments, Training)

Symptoms

- Click button, nothing happens
- Form doesn't submit
- Console shows JavaScript errors

Causes

- Fetch API call failing
- Backend endpoint not running
- JavaScript event handler missing
- Stale form data

Solution (Step-by-Step)

Step 1: Verify Backend is Running

Check Flask terminal for:

```
 * Running on http://127.0.0.1:5000
 * Press CTRL+C to quit
```

If not running:

```
python webapp.py
```

Step 2: Open Browser Console (F12)

Look for errors like:

```
GET http://127.0.0.1:5000/api/assignments 404 (Not Found)
TypeError: Cannot read property 'json' of undefined
```

Step 3: Check Backend Logs

Flask terminal should show requests:

```
127.0.0.1 - - [03/Dec/2025 14:25:33] "GET /api/assignments HTTP/1.1" 200
```

If you don't see request, button click isn't reaching backend.

Step 4: Hard Refresh Page

```
Ctrl + Shift + R  # Windows
Cmd + Shift + R   # macOS
```

Step 5: Try Different Button

If only one button broken, may be specific JavaScript issue.

Try another page to isolate problem.

Step 6: Test Endpoint Directly

Use curl to test API:

```
# Test assignment endpoint
curl http://127.0.0.1:5000/api/assignments

# Should return JSON data, not error
```

Prevention

- Monitor console errors during testing
- Test forms after code changes
- Verify all backend endpoints exist before frontend calls

6.3 React/Jinja2 Template Conflict

Symptoms

- Template syntax errors
- React variables not rendering
- `{{ }}` appearing in HTML instead of being processed

Causes

- Jinja2 tries to process React template variables
- Flask interprets `{{ }}` as Jinja2, not React

Solution

Wrap React code in `{% raw %}` tags:

```
<!-- In Flask template -->
<html>
  <body>
    {% raw %}
    <div id="root">
      <!-- React code here, Jinja2 won't process {{ }} -->
      const App = () => {
        return <h1>{{ title }}</h1>;
      }
    </div>
    {% endraw %}
  </body>
</html>
```

Prevention

- Use separate files for React and Jinja2 templates
- Use different syntax for each language
- Clearly comment which language each section uses

7.0 Functionality Issues

7.1 Session Timeout Happens Too Quickly

Symptoms

- Logged out after a few minutes of inactivity
- Users think it's a bug
- "Your session has expired" message appears

Causes

- This is **expected behavior** in demo mode
- SecureMed configured with 2-minute timeout for security demonstration
- Aligns with HIPAA §164.312(a)(2)(iii) automatic logoff requirement

Why This Happens

The 2-minute timeout demonstrates proper HIPAA security controls:

- Prevents unattended workstation access
- Forces users to re-authenticate
- Shown in educational/demo context

In production: Timeout should be 15-30 minutes, configurable per organization policy

Solution

For Demo/Testing Purposes: This is correct behavior - no action needed

To Increase Timeout (not recommended for demos):

Edit `webapp.py`:

```
from datetime import timedelta

# Find this line:
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=2)

# Change to:
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=30)
```

Then restart Flask:

```
python webapp.py
```

For Production Deployment:

```
# Make configurable via environment variable
timeout_minutes = int(os.getenv('SESSION_TIMEOUT_MINUTES', '30'))
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=timeout_minutes)
```

Documentation

Add to user guide:

```
Session Timeout:
- Demo environment: 2 minutes (security demonstration)
- 90-second warning before timeout
- Production: Configurable (typical 15-30 minutes)
- Aligns with HIPAA §164.312(a)(2)(iii) workstation security
```

Prevention

- Document timeout behavior in installation guide
- Show warning message at 90 seconds
- Explain this is intentional security feature

7.2 Training Score Not Updating

Symptoms

- Score stuck at 0 or previous value
- Final score shows incorrect value
- Score not persisting after logout

Causes

- Training answer not submitted correctly
- Database not committing changes
- JavaScript error preventing submission
- Not completing all questions

Solution (Step-by-Step)

Step 1: Complete All Questions

Training requires completion of all 3 modules (9 total questions).

Incomplete training won't show final score.

Step 2: Verify Each Answer Submits

After answering each question:

- Should see feedback (correct/incorrect)
- Should see point value (+20 or 0)
- Should advance to next question

If stuck on a question, hard refresh:

```
Ctrl + Shift + R  # Windows
Cmd + Shift + R   # macOS
```

Step 3: Check Browser Console

Open DevTools (F12) and look for errors when submitting answers.

Common errors:

```
POST http://127.0.0.1:5000/api/training/submit 500 (Server Error)
TypeError: Cannot read property 'score' of undefined
```

Step 4: Restart Backend

Database may not be committing:

```
# Stop Flask
Ctrl + C

# Restart Flask
python webapp.py

# Try training again
```

Step 5: Verify in Database

Advanced troubleshooting - check if score is actually saved:

```
# Check database (SQLite command line)
sqlite3 securemed.db

# Query:
SELECT username, training_score FROM users WHERE username='stefan';

# Should show your score
```

Prevention

- Don't close browser during training
- Complete all questions in one session
- Wait for each submission to complete before proceeding

7.3 Tasks Not Completing (Wrong Directory Entry)

Symptoms

- Task submission fails with error
- "Incorrect recipient" message shown
- Task marked as failed, violation created

Causes

- Directory code doesn't match exactly
- Typo in submission
- Using wrong recipient
- Directory mismatch between frontend and backend

Solution (Step-by-Step)

Step 1: Read Task Description Carefully

Task specifies exact recipient:

```
"Send secure message to Dr. Sarah Chen for patient John Doe"
```

You need to find "Dr. Sarah Chen" in the directory.

Step 2: Open Directory

Click "View Directory" or similar button.

Look for exact match:

```
Dr. Sarah Chen: SM-1847
```

Step 3: Check for Variations

Directory code might be:

- Case-insensitive: "SM-1847" = "sm-1847" □
- No spaces: "SM-1847" not "SM - 1847" □
- Full match required: "SM-1847" not "SM-18" □

Step 4: Enter Code Carefully

Copy-paste if possible:

```
Select code: SM-1847
Right-click → Copy
Click input field
Right-click → Paste
```

Or type carefully:

```
S M hyphen 1 8 4 7
```

Step 5: Verify Before Submitting

Double-check code matches directory exactly before clicking Submit.

Step 6: Submit

Click "Complete Task" or "Submit Assignment"

Expected Results:

- □ Correct: Green message "Task completed successfully"
- □ Incorrect: Red message "Incorrect recipient selected" + Violation created

Understanding Violations

Wrong submission creates violation (this is correct behavior):

```
Violation Type: Wrong task submission
Description: Selected SM-1848 (Dr. James Wilson) instead of SM-1847 (Dr. Sarah Chen)
User: stefan
Timestamp: 2025-12-03 14:30:00
```

This teaches proper PHI transmission practices.

Prevention

- Always use directory lookup before submission
- Match recipient name to directory code
- Don't guess at codes
- Take time to verify before submitting

7.4 Patient Edit Not Saving

Symptoms

- Edit patient info, click Save
- Success message appears
- Reload page, changes are gone

Causes

- Database transaction not committed
- Network error during save
- Permission issue
- Backend error not shown in UI

Solution

Step 1: Check Browser Console

Open DevTools (F12) → Console tab

Look for errors when saving:

```
POST http://127.0.0.1:5000/api/patients/update 500 (Server Error)
```

Step 2: Check Backend Logs

Look at Flask terminal for error messages:

```
[03/Dec/2025 14:30:00] "POST /api/patients/update HTTP/1.1" 500
```

Step 3: Verify Edit Permissions

Check that you're editing allowed fields:

- □ Editable: Email, Phone, Address
- □ Protected: MRN, Name, DOB, SSN

If trying to edit protected field, save won't work (expected).

Step 4: Try Editing Again

- Clear edit fields
- Re-enter new value
- Click Save again

Step 5: Restart Backend if Failing

```
# Stop
Ctrl + C

# Restart
python webapp.py

# Try editing again
```

Verification

After saving successfully:

1. Message shows: "Patient updated successfully"
2. Go to Audit Trail (Admin)
3. Filter by patient name
4. See `PATIENT_INFO_UPDATED` entry with changes

Example:

```
User: stefan
Action: PATIENT_INFO_UPDATED
Patient: John Doe (MRN2871)
Details: phone: '555-0101' → '555-0199'
```

Prevention

- Only edit allowed fields (email, phone, address)
- Wait for success message before navigating away
- Verify change in audit trail after saving

8.0 Security & Scanning Issues

8.1 Scanner Not Detecting Vulnerabilities

Symptoms

- EDR panel shows "No vulnerabilities detected"
- Want to test vulnerability scanning
- Scanner seems inactive

Causes

- No vulnerabilities configured in system
- Scanner not initialized
- Quick Setup not run

Solution

Step 1: Generate Demo Data

As **admin**, click "⚡ Quick Setup"

This auto-generates:

- 10-15 sample patients
- 5-10 sample vulnerabilities
- 3-5 sample violations

Step 2: Navigate to EDR Panel

Go to: **EDR** or **Threat Detection** section

Step 3: Verify Vulnerabilities Appear

Should see list like:

```
 CRITICAL: HTTPS Not Enabled
 HIGH: Missing Multi-Factor Auth
 MEDIUM: Outdated Dependency
```

Step 4: Test Remediation

Click "Mark Resolved" on a vulnerability

Verify it changes status to "RESOLVED"

If Still No Vulnerabilities

Option 1: Check Database

Advanced - verify vulnerabilities are in database:

```
sqlite3 securemed.db
SELECT * FROM vulnerabilities;
```

If empty, run Quick Setup again.

Option 2: Rebuild Scanner Data

Admin panel should have option to:

- Refresh vulnerability scan
- Regenerate sample vulnerabilities

Option 3: Manual Entry

Admin can manually add vulnerabilities via database.

Prevention

- Always run Quick Setup before testing EDR features
- Verify vulnerabilities loaded before testing remediation

8.2 Audit Logs Missing Entries

Symptoms

- Some user actions not appearing in audit log
- Logs seem incomplete
- Can't find specific activity

Causes

- Backend not restarted after code changes
- Database transaction not committed
- Audit logging disabled for some endpoints
- Log entries purged

Solution

Step 1: Restart Backend

Stop and restart Flask:

```
# Stop
Ctrl + C

# Restart
python webapp.py
```

Step 2: Perform Action Again

Do the action you want logged:

- Login
- Edit patient
- Submit training answer

Step 3: Check Audit Trail

Go to: **Audit Trail** (Admin only)

Filter for your action:

- Filter by username
- Filter by action type
- Check timestamp

Step 4: Verify Completeness

Test entry should show:

```
User: (your username)
Action: (action type, e.g., PATIENT_ACCESSED)
Timestamp: (current time)
Details: (details of action)
```

Advanced Verification

Check completeness test:

```
# Count audit entries
sqlite3 securemed.db "SELECT COUNT(*) FROM activity_log;"

# Should match number of actions performed
```

Prevention

- Restart backend after code changes

- Monitor audit trail regularly
- Don't delete audit logs (append-only system)
- Verify logging on new endpoints

8.3 Encryption Not Working (SSN Still Visible)

Symptoms

- SSN shows as plain text in database
- Encryption seems disabled
- Can read encrypted values with text editor

Causes

- Encryption function not called
- Encryption key missing
- Database not using encryption for this field

Solution

Step 1: Verify Encryption Key Exists

Check `webapp.py` for encryption key:

```
# Should have:
encryption_key = "..." # Fernet key
```

If missing or commented out, encryption disabled.

Step 2: Test Encryption

Create test patient and check database:

```
sqlite3 securemed.db
SELECT ssn FROM patients LIMIT 1;
```

If SSN is encrypted:

- Starts with `gAAAAAA...` (base64 encoded)
- Not readable as plain text □

If SSN is plain text:

- Shows as `123-45-6789`
- Something is wrong □

Step 3: Enable Encryption

If encryption disabled, uncomment or add to `webapp.py`:

```
from cryptography.fernet import Fernet

# Generate or load encryption key
encryption_key = Fernet.generate_key()  # Or load from secure storage

cipher = Fernet(encryption_key)
```

Step 4: Re-encrypt Existing Data

For existing plain-text SSNs, run encryption script (if available):

```
python encrypt_existing_data.py
```

Or clear database and re-add:

```
rm securemed.db
python webapp.py  # Creates new encrypted database
```

Verification

After fix:

```
sqlite3 securemed.db
SELECT ssn FROM patients WHERE mrn='MRN00001';
```

Should show encrypted string starting with `gAAAAAA`.

Security Note

⚠ If SSN ever appears in plain text, this is a HIPAA violation:

- §164.312(a)(2)(iv) requires encryption
- Audit who accessed unencrypted SSN
- Document breach assessment

9.0 PDF Generation Issues

9.1 PDF Not Generating (or Downloads Blank)

Symptoms

- Click "Generate Report" button
- File downloads but is blank or corrupted
- No error message shown

Causes

- ReportLab not installed
- Insufficient disk space
- Permission issues
- Database connection lost

Solution

Step 1: Verify ReportLab Installed

```
# Check if installed
pip show reportlab

# Should show: Version 4.4.4

# If not installed:
pip install reportlab==4.4.4

# Restart Flask
python webapp.py
```

Step 2: Check Disk Space

```
# Check available space
df -h     # macOS/Linux
dir       # Windows

# Need at least 100 MB free for PDF generation
```

Step 3: Test PDF Generation

Try generating small report:

- Go to Admin dashboard
- Click "Generate Report"
- Select smallest report type
- Try generating

Step 4: Check Backend Logs

Flask terminal should show:

```
[03/Dec/2025 14:35:00] POST /api/reports/generate HTTP/1.1 200
```

If you see 500 error, check error message in terminal.

Step 5: Verify Database Connection

Backend needs to query database for report data:

```
# Ensure database file exists and is accessible
ls -la securemed.db

# Should show file with good permissions
```

If Still Failing

Option 1: Manual PDF Generation

Advanced - use Python directly:

```
# In Python
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

c = canvas.Canvas("test.pdf", pagesize=letter)
c.drawString(100, 750, "Test PDF")
c.save()

# Check if test.pdf created successfully
```

Option 2: Reinstall ReportLab

```
pip uninstall reportlab
pip install reportlab==4.4.4 --force-reinstall
```

Prevention

- Keep disk space >500 MB
- Verify ReportLab after installation
- Test PDF generation regularly

10.0 Browser & Client Issues

10.1 Login Page Doesn't Load (Blank Page)

Symptoms

- Go to `http://127.0.0.1:5000/login`
- Page stays blank
- No content appears

Causes

- Flask not serving static files
- Browser cache issues
- JavaScript loading issue
- CSS not loaded

Solution

Step 1: Check Flask Terminal

Should show request logged:

```
127.0.0.1 - - [03/Dec/2025 14:20:00] "GET /login HTTP/1.1" 200
```

If you see 404 instead of 200, template not found.

Step 2: Hard Refresh Browser

Clear cache and reload:

```
Ctrl + Shift + Delete    # Clear cache
Ctrl + F5               # Hard refresh
```

Then refresh page.

Step 3: Try Different Browser

Test in different browser (Chrome, Firefox, Safari, Edge):

- If works in one, issue is browser-specific
- Try clearing cache in your original browser

Step 4: Check Console

Open DevTools (F12) → Console tab

Look for errors like:

- 404 File not found
- CORS blocked
- Template error

Step 5: Verify Template File

Backend should be serving login.html:

```
ls templates/login.html

# Should exist and be readable
```

10.2 Login Credentials Not Working

Symptoms

- Enter username and password
- Get "Invalid username or password" error
- Can't log in

Causes

- Wrong credentials entered
- Caps Lock on (username/password case-sensitive)
- Credentials table corrupted
- Demo accounts not created

Solution

Step 1: Verify Default Credentials

Use official demo accounts:

Username	Password
admin	Admin123!
stefan	Stefan123!
ana	Ana123!

Note the exact case - passwords are case-sensitive.

Step 2: Check Caps Lock

Verify Caps Lock is OFF while typing password.

Step 3: Clear Username/Password Fields

- Select all and delete
- Type slowly and carefully
- Use copy-paste if available

Step 4: Verify Backend is Running

Flask should show:

```
 * Running on http://127.0.0.1:5000
```

If Flask crashed, restart it.

Step 5: Reset Demo Accounts

If demo accounts corrupted:

```
# Stop Flask
Ctrl + C

# Delete database
rm securemed.db

# Restart Flask (recreates with default accounts)
python webapp.py
```

10.3 Button Clicks Not Working in Login Form

Symptoms

- Can type in username/password
- Click "Login" button
- Nothing happens

Causes

- JavaScript error in form
- Form action not configured
- Fetch API request failing

Solution

Step 1: Open Console (F12)

Look for JavaScript errors.

Step 2: Check Network Tab

In DevTools Network tab:

- Click Login button
- Watch for POST request to `/login`
- Check if request succeeds (200) or fails (500)

Step 3: Try Enter Key

Instead of clicking Login button:

- Type credentials
- Press Enter key
- Sometimes submits when button click doesn't

Step 4: Check Browser Compatibility

Try in different browser.

10.4 Styles Not Applied (Page Looks Broken)

Symptoms

- Page loads but no styling
- Text all same size/color
- Layout looks broken

Causes

- CSS file not loaded
- Static files not served
- File permissions issue

Solution

Step 1: Check Static Folder

```
ls static/
# Should show: style.css and other files
```

Step 2: Verify Flask Serving Static Files

In `webapp.py`, should have:

```
app = Flask(__name__,
            template_folder='templates',
            static_folder='static')
```

Step 3: Check File Permissions

```
# macOS/Linux
ls -la static/style.css

# Should be readable (r at start of permissions)
```

Step 4: Hard Refresh

```
Ctrl + Shift + R   # Clear styles from cache
```

11.0 Network & Connectivity Issues

11.1 Cannot Access http://127.0.0.1:5000 (http://127.0.0.1:5000)

Symptoms

- Browser shows "Cannot reach server" or "Connection refused"
- Page won't load

Causes

- Flask not running
- Wrong port number
- Firewall blocking
- localhost resolution issue

Solution

Step 1: Verify Flask is Running

Terminal where you started Flask should show:

```
 * Running on http://127.0.0.1:5000
 * Press CTRL+C to quit
```

If not, start it:

```
python webapp.py
```

Step 2: Check Port

If port 5000 not available, Flask shows different port:

```
 * Running on http://127.0.0.1:5001
```

Use that address in browser.

Step 3: Test Connectivity

```
curl http://127.0.0.1:5000/login

# Should return HTML for login page
```

Step 4: Check Firewall

Windows firewall might block port 5000:

- Allow Python through firewall
- Or use different port (5001, 5002, etc.)

Step 5: Try localhost instead of IP

If `127.0.0.1` doesn't work, try:

```
http://localhost:5000/login
```

11.2 Slow Response Times

Symptoms

- Pages load very slowly (>5 seconds)
- API requests take long time
- System feels sluggish

Causes

- Disk I/O bottleneck
- Database queries slow
- Encryption overhead
- System resource constraints

Solution

Step 1: Monitor System Resources

```
# Check CPU, memory, disk
top       # macOS/Linux
Task Manager  # Windows
```

If at capacity, close other applications.

Step 2: Check Database Size

```
# Get database file size
ls -lh securemed.db

# If >1 GB, consider archiving old data
```

Step 3: Optimize Flask Settings

In `webapp.py`:

```
# Disable debug mode for faster performance
app.run(debug=False, port=5000)  # Change from debug=True
```

Restart Flask.

Step 4: Add Database Indexes

Advanced - optimize slow queries:

```
cursor.execute("CREATE INDEX idx_patient_mrn ON patients(mrn);")
```

12.0 General Debugging Tips

12.1 Always Check These First

1. **Is Flask running?**

   ```
   # Check terminal where you started Flask
   # Should show: Running on http://127.0.0.1:5000
   ```

2. **Is virtual environment activated?**

```
# Prompt should show (venv) at start
(venv) user@computer Cap_Finaldev %
```

3. **Are you in correct directory?**

```
pwd     # macOS/Linux
cd      # Windows
# Should show /path/to/Cap_Finaldev
```

4. **Is database file present?**

```
ls -la securemed.db
# Should exist and show recent timestamp
```

5. **Check browser console (F12)**
    - Look for red error messages
    - Check Network tab for failed requests

12.2 Systematic Debugging Approach

When something breaks:

1. **Identify the problem** - What exactly isn't working?
2. **Reproduce the issue** - Can you make it happen consistently?
3. **Check logs** - Flask terminal and browser console
4. **Isolate the cause** - Is it backend or frontend?
5. **Test fix** - Try solution one at a time
6. **Verify resolution** - Confirm problem is solved
7. **Document** - Note what fixed it for future reference

12.3 Useful Debugging Commands

Python/Flask:

```
# Test Python syntax
python -m py_compile webapp.py

# Test Flask import
python -c "from flask import Flask; print('OK')"

# Run Flask with verbose output
python -u webapp.py  # Unbuffered, shows all output
```

Database:

```
# Interactive SQLite shell
sqlite3 securemed.db

# Inside shell:
SELECT * FROM users;              # List users
SELECT COUNT(*) FROM activity_log; # Count audit entries
```

Network:

```
# Test if port is in use
lsof -i :5000                     # macOS/Linux
netstat -ano | findstr :5000      # Windows

# Test connectivity
curl http://127.0.0.1:5000/login
```

Process:

```
# Find Flask processes
ps aux | grep webapp.py           # macOS/Linux
tasklist | findstr python.exe     # Windows

# Kill process
kill -9 <PID>                     # macOS/Linux
taskkill /PID <PID> /F            # Windows
```

12.4 Enable Debug Mode for Development

To see more detailed error messages:

In `webapp.py`:

```
if __name__ == '__main__':
    app.run(debug=True)  # Shows detailed errors
```

Then access in browser to see full error traceback.

⚠ **Important**: Never use `debug=True` in production - it's a security risk.

13.0 When to Seek Help

13.1 Issues You Can Solve Yourself

 Recommended: Try troubleshooting for 15-30 minutes first

These are typically self-service issues:

- ModuleNotFoundError - reinstall dependencies
- Port already in use - kill process
- Template not found - change directory
- Database locked - restart Flask
- Login credentials - verify spelling
- Session timeout - expected behavior

13.2 When to Ask for Help

 Ask your instructor/team if:

- Problem persists after 30 minutes troubleshooting
- Multiple issues occurring simultaneously
- System crashes repeatedly
- Data appears corrupted
- Security concern suspected
- Error messages are unclear

When asking for help, provide:

1. Exact error message (copy from terminal)
2. What you were doing when error occurred
3. Steps you've already tried
4. Output of `pip list` (installed packages)
5. Flask terminal output showing the error

13.3 Emergency Support

Critical Issues:

- Data loss / corruption
- Security vulnerability
- Complete system failure

Action:

1. Stop using system immediately
2. Contact instructor/admin with full details
3. Don't attempt to fix - may worsen situation

14.0 Conclusion

This troubleshooting guide covers the most common issues encountered with SecureMed. Most problems can be resolved using the solutions provided in this document.

Key Takeaways

1. **Always check the basics first** - Flask running, venv activated, correct directory
2. **Read error messages carefully** - They usually tell you the problem
3. **Restart Flask** - Solves 50% of issues
4. **Monitor logs** - Flask terminal and browser console show what's happening
5. **Test methodically** - Change one thing at a time, then test

Preventive Maintenance

To avoid most issues:

- Always activate virtual environment
- Never move project folders
- Regularly backup `securemed.db`
- Keep Flask running properly between sessions
- Check browser console for errors
- Document what worked for troubleshooting in future

Quick Recap of Top 5 Fixes

1. **ModuleNotFoundError**: `source venv/bin/activate` + `pip install -r requirements.txt`
2. **Port in use**: `lsof -ti:5000 | xargs kill -9`
3. **Template not found**: `cd /path/to/Cap_Finaldev`
4. **Database locked**: Stop Flask (`Ctrl+C`) and restart
5. **Nothing working**: `Ctrl+C`, delete `securemed.db`, restart Flask

Next Steps

If you've solved your issue:

- Return to using SecureMed normally
- Document solution for team (improves efficiency)
- Test thoroughly to prevent future issues

If issue persists:

- Review this guide again for similar issues
- Provide detailed information to your instructor
- Work with team to resolve systematically

Document Information:

- **Version**: 1.0 - Final
- **Last Updated**: December 2025
- **Scope**: Covers issues found during Sprints 3-6 testing
- **Audience**: SecureMed users, developers, administrators
- **Institution**: Florida International University
- **Course**: CIS 4914 - Cybersecurity Capstone Project II

Quick Help Shortcuts:

Problem	Solution	Command
Flask not starting	Install deps	`pip install -r requirements.txt`
Port busy	Kill process	`lsof -ti:5000 | xargs kill -9`
Wrong directory	Navigate	`cd /path/to/Cap_Finaldev`
DB locked	Restart	`Ctrl+C` then `python webapp.py`
Need new DB	Reset	`rm securemed.db` then restart

Remember: Most issues are solved by restarting Flask with `Ctrl+C` then `python webapp.py`

If all else fails, use the Emergency Reset in section 2.3!