Manual Test Checklist - CSC255 Certificate Project

Date: December 4, 2025
Application URL: http://127.0.0.1:5000

Pre-Test Setup

- [x] Python 3.8+ installed
- [x] Dependencies installed: `pip install -r requirements.txt`
- [x] Flask application started: `python app.py`
- [x] Application accessible at http://127.0.0.1:5000
- [x] Browser opened and ready

Test Flow 1: Complete Happy Path (Generate → Download → Upload → Success)

Step 1: Navigate to Home Page
- [ ] Open http://127.0.0.1:5000 in browser
- [ ] Verify home page loads successfully
- [ ] Verify "Generate Keys & Certificate" button is visible
- [ ] Verify "Authenticate Certificate" button is visible

Expected Result: Home page properly opens with navigation options
Status: Pass

Step 2: Generate Keys and Certificate
- [ ] Click "Generate Keys & Certificate" button
- [ ] Wait for generation to complete
- [ ] Verify redirect to generation results page
- [ ] Verify three download links appear:
  - [ ] Download Private Key
  - [ ] Download Public Key
  - [ ] Download Certificate
- [ ] Verify no error messages displayed

Expected Result: Three download links available, no errors
Status: Pass

Step 3: Download Certificate File
- [ ] Click "Download Certificate" link
- [ ] Verify file download begins
- [ ] Verify downloaded file is named `certificate.pem`
- [ ] Verify file size is reasonable (not empty, not too large)
- [ ] Open file in text editor
- [ ] Verify file contains:

  - [ ] `-----BEGIN CERTIFICATE-----` header
  - [ ] Base64 encoded content
  - [ ] `-----END CERTIFICATE-----` footer

Expected Result: Valid PEM certificate file downloaded
Status: Pass

Step 4: Navigate to Authentication Page
- [ ] Click "Authenticate Certificate" button (or navigate back to home and click it)
- [ ] Verify authentication page loads
- [ ] Verify file upload form is visible
- [ ] Verify "Choose File" button is present
- [ ] Verify "Upload and Verify" button is present

Expected Result: Clean upload form ready for file selection
Status: Pass

Step 5: Upload Valid Certificate
- [ ] Click "Choose File" button
- [ ] Navigate to downloaded `certificate.pem` file
- [ ] Select the file
- [ ] Verify filename appears next to "Choose File" button
- [ ] Click "Upload and Verify" button
- [ ] Wait for processing

Expected Result: File selected and upload initiated
Status: Pass

Step 6: Verify Success Result
- [ ] Verify redirect to success page
- [ ] Verify success message displayed (e.g., "Certificate is valid!")
- [ ] Verify green/positive styling
- [ ] No error messages displayed

Expected Result: Success page confirming valid certificate
Status: Pass

Test Flow 2: Invalid Certificate (Generate → Upload Different → Fail)

Step 1: Create Invalid Certificate File
- [ ] Create a new text file named `invalid_cert.pem`
- [ ] Add content:
  ```
  -----BEGIN CERTIFICATE-----
  ```

```
  InvalidCertificateContent123
  -----END CERTIFICATE-----
```
- [ ] Save file

Expected Result: Invalid certificate file created
Status: Pass

Step 2: Upload Invalid Certificate
- [ ] Navigate to authentication page
- [ ] Click "Choose File"
- [ ] Select `invalid_cert.pem`
- [ ] Click "Upload and Verify"
- [ ] Wait for processing

Expected Result: File uploaded for verification
Status: Pass

Step 3: Verify Failure Result
- [ ] Verify redirect to failure page
- [ ] Verify failure message displayed (e.g., "Certificate verification failed")
- [ ] Verify red/negative styling
- [ ] Verify option to try again

Expected Result: Failure page confirming invalid certificate
Status: Pass

Test Flow 3: File Type Validation

Step 1: Upload Non-PEM File
- [ ] Create a text file named `test.txt`
- [ ] Navigate to authentication page
- [ ] Attempt to upload `test.txt`

Expected Result: Error message about invalid file type
Status: Pass

Step 2: Upload No File
- [ ] Navigate to authentication page
- [ ] Click "Upload and Verify" without selecting a file

Expected Result: Error message about no file selected
Status: Pass

Test Flow 4: Multiple Downloads

Step 1: Download All Generated Files
- [ ] Generate new keys and certificate
- [ ] Download private key → verify file `private_key.pem`
- [ ] Download public key → verify file `public_key.pem`
- [ ] Download certificate → verify file `certificate.pem`
- [ ] Verify all three files have appropriate content

Expected Result: All three files download successfully
Status: Pass

Step 2: Verify File Contents
- [ ] Open `private_key.pem` - contains `BEGIN RSA PRIVATE KEY`
- [ ] Open `public_key.pem` - contains `BEGIN PUBLIC KEY`
- [ ] Open `certificate.pem` - contains `BEGIN CERTIFICATE`

Expected Result: All files have proper PEM format
Status: Pass

Test Flow 5: Large File Upload (Security Test)

Step 1: Create Large File
- [ ] Create a file larger than 1MB
- [ ] Name it `large_cert.pem`

Expected Result: Test file created
Status: Pass

Step 2: Attempt Upload
- [ ] Navigate to authentication page
- [ ] Attempt to upload large file

Expected Result: Error message about file size limit (1MB max)
Status: Pass

Test Flow 6: Performance Testing

Test 1: Multiple Rapid Generations
- [ ] Generate certificate 5 times in quick succession
- [ ] Verify no errors or crashes

Expected Result: All generations succeed
Status: Pass

Test 2: Concurrent Users (if possible)
- [ ] Open application in multiple browser tabs
- [ ] Generate certificates simultaneously
- [ ] Verify no session collision

Expected Result: Each tab operates independently
Status: Pass

Test Summary

Total Tests: 17
Tests Passed: __17_
Tests Failed: _0__
Tests Skipped: __0_
Pass Rate: _100__%

Critical Issues Found: N/A

Minor Issues Found: N/A

Recommendations: Implement automatic cleanup of old temp files in the temp directory (currently files accumulate indefinitely). Consider adding HTTPS support for production deployment since program is handling sensitive cryptographic materials.