**Test Results Report**

**Test Results Overview**

Based on the comprehensive test suite for the ActiveSky project, we conducted multiple acceptance tests across key functional areas. Here's a detailed breakdown of the test results:

1. User Login Tests

- test\_user\_login\_success : PASSED

- Validates successful login with correct credentials

- Confirms user can access the homepage after authentication

- test\_user\_login\_wrong\_password : PASSED

- Confirms system rejects login attempts with incorrect passwords

- Ensures password security mechanisms are working

- test\_user\_login\_nonexistent\_email : PASSED

- Verifies system handles attempts to log in with non-existent email addresses

- Demonstrates robust error handling in authentication process

2. User Registration Tests

- test\_user\_registration\_success : PASSED

- Validates successful user registration with valid credentials

- Confirms user account creation process works as expected

- test\_user\_registration\_existing\_email : PASSED

- Ensures system prevents registration with an email already in use

- Demonstrates email uniqueness validation

- test\_user\_registration\_short\_email : PASSED

- Confirms email length validation works

- Prevents registration with inadequately short email addresses

- test\_user\_registration\_short\_password : PASSED

- Validates password length requirements

- Ensures minimum password complexity standards are enforced

3. API Integration Tests

- test\_weather\_api\_response : PASSED

- Confirms successful retrieval of weather data from OpenWeatherAPI

- Validates API integration and data parsing

4. Recommendation Generation Tests

- test\_time\_based\_recommendations : PASSED

- Verifies system generates recommendations based on time

- Demonstrates dynamic recommendation adaptation

- test\_temperature\_recommendations : PASSED

- Confirms recommendations vary with temperature changes

- Shows temperature-sensitive recommendation logic

- test\_weather\_condition\_recommendation : PASSED

- Validates recommendations adapt to different weather conditions

- Demonstrates complex recommendation generation algorithm

Performance and Compatibility Tests

1. Page Loading Performance

- test\_home\_page\_loading\_time : PASSED

- Home page loads in under 2 seconds

- test\_login\_page\_loading\_time : PASSED

- Login page loads in under 2 seconds

- test\_signup\_page\_loading\_time : PASSED

- Signup page loads in under 2 seconds

2. Compatibility Test

- test\_os\_compatibility : PASSED

- Confirms application supports multiple operating systems (Linux, Windows, Darwin(macOs) )

**Test Results Analysis**

**Key Observations**

- All defined acceptance tests passed successfully

- No critical failures were encountered during testing

- The system demonstrates robust handling of various scenarios including:

- User authentication

- Registration validation

- Weather API integration

- Dynamic recommendation generation

**Potential Improvements**

While all tests passed, potential areas for future enhancement include:

- Implementing more granular password complexity rules

- Adding more diverse recommendation scenarios

- Expanding cross-platform compatibility testing

## Detailed Test Logs

### Deployment Logs

Received test ids from temp file.

test\_temperature\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_temperature\_recommendations)

Test if temperature-based activity recommendations work correctly ... ok

test\_time\_based\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_time\_based\_recommendations)

Test if time-based activity recommendations work correctly ... ok

test\_weather\_condition\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_weather\_condition\_recommendations)

Test if weather condition-based activity recommendations work correctly ... ok

test\_os\_compatibility (os\_compatibility\_test.OperatingSystemTest.test\_os\_compatibility)

Test if the current operating system is supported ... Created Database!

ok

Created Database!

test\_home\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_home\_page\_loading\_time)

Test if the homepage loads in less than 2 seconds ... ok

Created Database!

test\_login\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_login\_page\_loading\_time)

Test if the login page loads in less than 2 seconds ... ok

Created Database!

test\_signup\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_signup\_page\_loading\_time)

Test if the signup page loads in less than 2 seconds ... ok

test\_temperature\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_temperature\_recommendations)

Test if temperature-based activity recommendations work correctly ... ok

test\_time\_based\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_time\_based\_recommendations)

Test if time-based activity recommendations work correctly ... ok

test\_weather\_condition\_recommendations (activity\_recommendation\_test.ActivityRecommendationTest.test\_weather\_condition\_recommendations)

Test if weather condition-based activity recommendations work correctly ... ok

test\_os\_compatibility (os\_compatibility\_test.OperatingSystemTest.test\_os\_compatibility)

Test if the current operating system is supported ... Created Database!

ok

Created Database!

test\_home\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_home\_page\_loading\_time)

Test if the homepage loads in less than 2 seconds ... ok

Created Database!

test\_login\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_login\_page\_loading\_time)

Test if the login page loads in less than 2 seconds ... ok

Created Database!

test\_signup\_page\_loading\_time (page\_loading\_test.PageLoadingTimeTest.test\_signup\_page\_loading\_time)

Test if the signup page loads in less than 2 seconds ... ok

Created Database!

test\_user\_login\_nonexistent\_email (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_nonexistent\_email)

Test if login fails with non-existent email ... ok

Created Database!

test\_user\_login\_success (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_success)

Test if a user can log in with correct credentials ... ok

Created Database!

test\_user\_login\_wrong\_password (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_wrong\_password)

Test if login fails with incorrect password ... ok

Created Database!

test\_user\_registration\_existing\_email (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_existing\_email)

Test if registration fails with existing email ... ok

Created Database!

test\_user\_registration\_short\_email (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_email)

Test if registration fails with short email ... ok

Created Database!

test\_user\_registration\_short\_password (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_password)

Test if registration fails with short password ... ok

Created Database!

test\_user\_registration\_short\_username (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_username)

Test if registration fails with short username ... ok

Created Database!

test\_user\_registration\_success (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_success)

Test if a new user can register successfully ... ok

Created Database!

test\_weather\_api\_response (weather\_api\_test.WeatherAPITest.test\_weather\_api\_response)

Test if the weather API endpoint returns valid data ... ok

Created Database!

test\_user\_login\_nonexistent\_email (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_nonexistent\_email)

Test if login fails with non-existent email ... ok

Created Database!

test\_user\_login\_success (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_success)

Test if a user can log in with correct credentials ... ok

Created Database!

test\_user\_login\_wrong\_password (user\_authentication\_test.UserAuthenticationTest.test\_user\_login\_wrong\_password)

Test if login fails with incorrect password ... ok

Created Database!

test\_user\_registration\_existing\_email (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_existing\_email)

Test if registration fails with existing email ... ok

Created Database!

test\_user\_registration\_short\_email (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_email)

Test if registration fails with short email ... ok

Created Database!

test\_user\_registration\_short\_password (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_password)

Test if registration fails with short password ... ok

Created Database!

test\_user\_registration\_short\_username (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_short\_username)

Test if registration fails with short username ... ok

Created Database!

test\_user\_registration\_success (user\_registration\_test.UserRegistrationTest.test\_user\_registration\_success)

Test if a new user can register successfully ... ok

Created Database!

test\_weather\_api\_response (weather\_api\_test.WeatherAPITest.test\_weather\_api\_response)

Test if the weather API endpoint returns valid data ... ok

----------------------------------------------------------------------

Ran 32 tests in 2.694s

OK

Finished running tests!

## Test Modules Executed

* activity\_recommendation\_test
* os\_compatibility\_test
* page\_loading\_test
* user\_authentication\_test
* user\_registration\_test
* weather\_api\_test

## Conclusion

The ActiveSky project demonstrates a robust, well-tested application with comprehensive validation across multiple functional areas. All 32 tests passed successfully, indicating a high-quality, reliable software implementation.