Software Testing Report – Soul of Bangladesh

Project: Soul of Bangladesh
Architecture: MVC (Model–View–Controller)
Testing Types Used: Unit Testing, Integration Testing

Introduction

To ensure *Soul of Bangladesh* is stable, modular, and user-ready, two structured testing methods were applied.
To keep the workflow clean and industry-aligned, **one dedicated tool per test type** was used instead of multiple overlapping frameworks.

Tools Selected

Testing Type	Tool Chosen	Purpose
Unit Testing	**PHPUnit**	Test individual components like controllers, models, functions
Integration Testing	**Codeception**	Test full system flows (DB, API, forms, login, UI behavior, modules working together)

Why These Tools?

Why PHPUnit for Unit Testing?

1. Industry standard for PHP apps
2. Perfectly fits MVC component testing
3. Supports mock objects
4. Fast, lightweight, and automation-friendly

Why Codeception for Integration Testing?

1. Built **specifically for integration & system testing**

2. Can test **DB, APIs, UI, Forms, Auth** together
3. Simulates real user behavior
4. Has database verification built-in
5. Best choice when modules must communicate properly

What Was Tested

Unit Testing (PHPUnit)

Component	**Purpose**
Controllers	Validate request handling
Models	Ensure correct DB queries
Helper functions	Validate logic & return values
Form validation	Check input processing

Integration Testing (Codeception)

Scenario	**Test Goal**
Login → Dashboard	Correct authentication flow
Story Submission	Save to DB & reflect on UI
Comments & Reactions	Multi-module interaction
Admin Panel	Restricted actions & DB updates
Full request cycles	Controller → Model → DB → Response

Project Testing Folder Structure

```
SoulOfBangladesh/
|— tests/
|   ├— Unit/              # PHPUnit tests
|   |   └— StoryTest.php
|   ├— Integration/       # Codeception tests
|   |   └— StoryCest.php
|— vendor/
|— app/
|— public/
|— codeception.yml
|— phpunit.xml
```

Sample Test Cases

PHPUnit – Unit Test Example

Testing model data insertion

```
use PHPUnit\Framework\TestCase;
use App\Models\StoryModel;

class StoryTest extends TestCase {
    public function testStoryInsert() {
        $story = new StoryModel();

        $result = $story->addStory("Liberation War", "This is a historic story");
```

```
        $this->assertTrue($result);
    }
}
```

Codeception – Integration Test Example

Testing full story submission flow

```
class StoryCest {
    public function submitStoryTest(IntegrationTester $I) {
        $I->amOnPage('/submit-story');
        $I->fillField('title', 'Language Movement 1952');
        $I->fillField('description', 'Historic moment for Bangladesh');
        $I->click('Submit');

        $I->seeInDatabase('stories', [
            'title' => 'Language Movement 1952'
        ]);

        $I->see('Story Submitted Successfully');
    }
}
```

Command Setup to Run Tests

▶ Run PHPUnit (Unit Tests)

```
./vendor/bin/phpunit tests/Unit
```

▶ Run Codeception (Integration Tests)

```
./vendor/bin/codecept run integration
```

Benefits Achieved in the Project

Benefit	Impact on Soul of Bangladesh
Bug-free components	No broken models/controllers
Stable system workflows	Smooth login, posts, admin actions
Secure database handling	Clean validated data
Real-world user simulation	Tests behave like real users
Scalable codebase	Easy to expand features

Conclusion

By using:

PHPUnit → to verify isolated components

Codeception → to validate full system behavior

The project reaches:

✅ Higher stability
✅ Better security
✅ Scalable architecture
✅ Real user flow validation
✅ Production-ready confidence

This testing strategy ensures *Soul of Bangladesh* is not just functional, but **engineering-solid**.