

Migration Tools & Decision Document

1. Introduction

This document outlines the various tools and options considered during the migration of an old legacy application to a modern Spring Boot-based architecture. It also explains the reasoning behind key technical choices.

2. Code Migration Strategy

Tool Considered: OpenRewrite

Purpose: Automated refactoring and migration (e.g., JEE to Spring Boot)

License: Open source (Apache 2.0) — free to use in enterprises

Ease of Use: Requires understanding of its YAML-based recipe structure; CLI and Gradle/Maven plugins available

Integration: Integrates well with build pipelines (Maven/Gradle) and CI tools

Customizability: Allows custom recipes for organization-specific migrations

Reason for Consideration:

- Supports large-scale code transformation
- Actively maintained and extensible
- Can detect outdated constructs and auto-update them

Outcome: Selected for reference; used manual migration with IDE support for smaller project, OpenRewrite for bulk refactoring can be used.

3. Automated Test Generation

Options Evaluated:

- **Diffblue Cover**
 - Pros: Fully automated JUnit generation, integration with IDE
 - Cons: Licensing cost, less flexible for custom pipelines

- **OpenAI Code Generation API**

- Pros: Customizable, generative AI for contextual test creation
- Cons: Requires integration effort, Requires prompt development

Comparative Analysis:

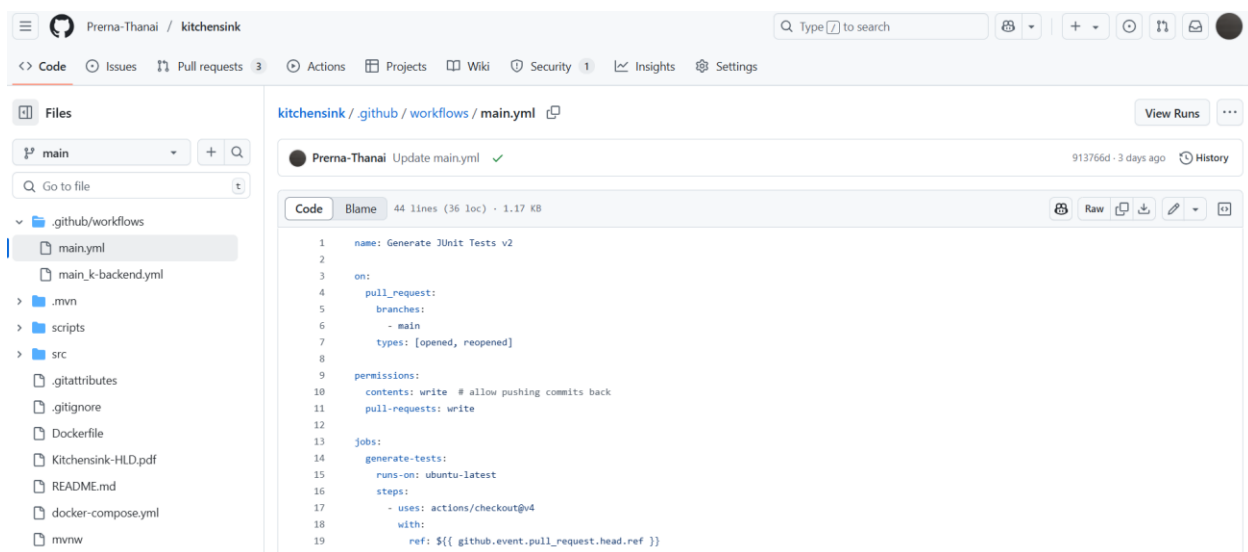
	Diffblue	OpenAI
License	Commercial	Pay-as-you-go
Integration	IDE plugin, pipeline integration	No IDE plugin, pipeline integration
Customizability	Low	High(via prompts)
Cost	\$30K per year (Teams)	\$13/1M token

Decision:

- Created a custom test-generation pipeline using OpenAI APIs which runs at every PR creation.
- Referenced Diffblue for test design structure and pipeline automation
- Can be customized further for specific use cases

Walkthrough:

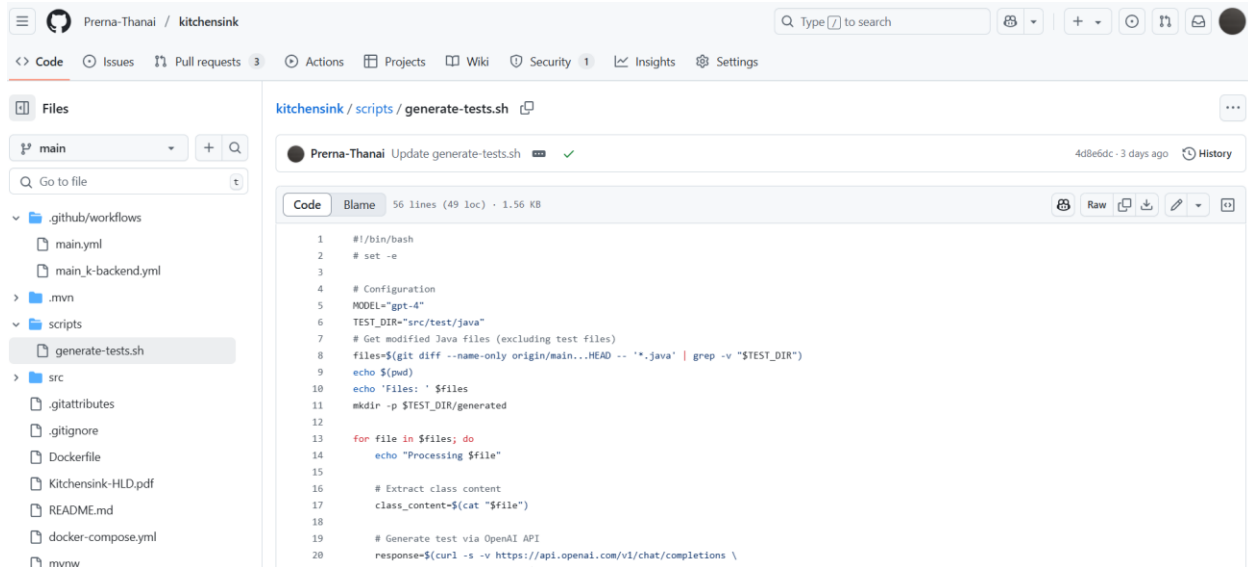
Created a pipeline using Github Actions to run at every PR creation.



The screenshot shows a GitHub repository interface for 'Prerna-Thandai / kitchensink'. The 'Code' tab is selected, displaying the workflow file 'main.yml' located at '.github/workflows/main.yml'. The workflow is titled 'Generate JUnit Tests v2' and is triggered on pull requests to the 'main' branch. It includes permissions for 'contents: write' and 'pull-requests: write'. The workflow consists of a single job named 'generate-tests' that runs on 'ubuntu-latest' and uses the 'actions/checkout@v4' action. The workflow file content is as follows:

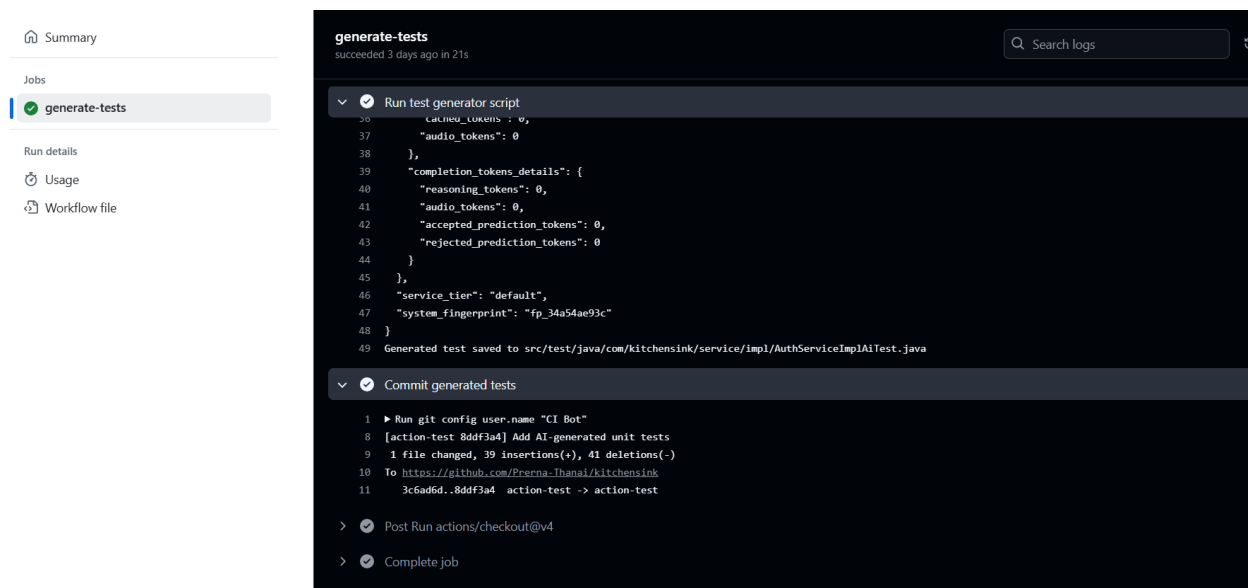
```
1 name: Generate JUnit Tests v2
2
3 on:
4   pull_request:
5     branches:
6       - main
7     types: [opened, reopened]
8
9 permissions:
10  contents: write # allow pushing commits back
11  pull-requests: write
12
13 jobs:
14   generate-tests:
15     runs-on: ubuntu-latest
16     steps:
17       - uses: actions/checkout@v4
18         with:
19           ref: ${{ github.event.pull_request.head.ref }}
```

A script is created which is executed by the pipeline. Script implements OpenAPI integration using the accurate prompts for test generation.



```
1 #!/bin/bash
2 # set -e
3
4 # Configuration
5 MODEL="gpt-4"
6 TEST_DIR="src/test/java"
7 # Get modified Java files (excluding test files)
8 files=$(git diff --name-only origin/main...HEAD -- '*.java' | grep -v "$TEST_DIR")
9 echo $(pwd)
10 echo 'Files: ' $files
11 mkdir -p $TEST_DIR/generated
12
13 for file in $files; do
14     echo "Processing $file"
15
16     # Extract class content
17     class_content=$(cat "$file")
18
19     # Generate test via OpenAI API
20     response=$(curl -s -v https://api.openai.com/v1/chat/completions \
21         -H "Content-Type: application/json" \
22         -d '{
23             "model": "'$MODEL'",
24             "messages": [
25                 {
26                     "role": "system",
27                     "content": "You are a test generator. Generate a test for the following class content."
28                 },
29                 {
30                     "role": "user",
31                     "content": "'$class_content'"
32                 }
33             ],
34             "max_tokens": 100,
35             "temperature": 0.5
36         }')
```

Pipeline Execution State –



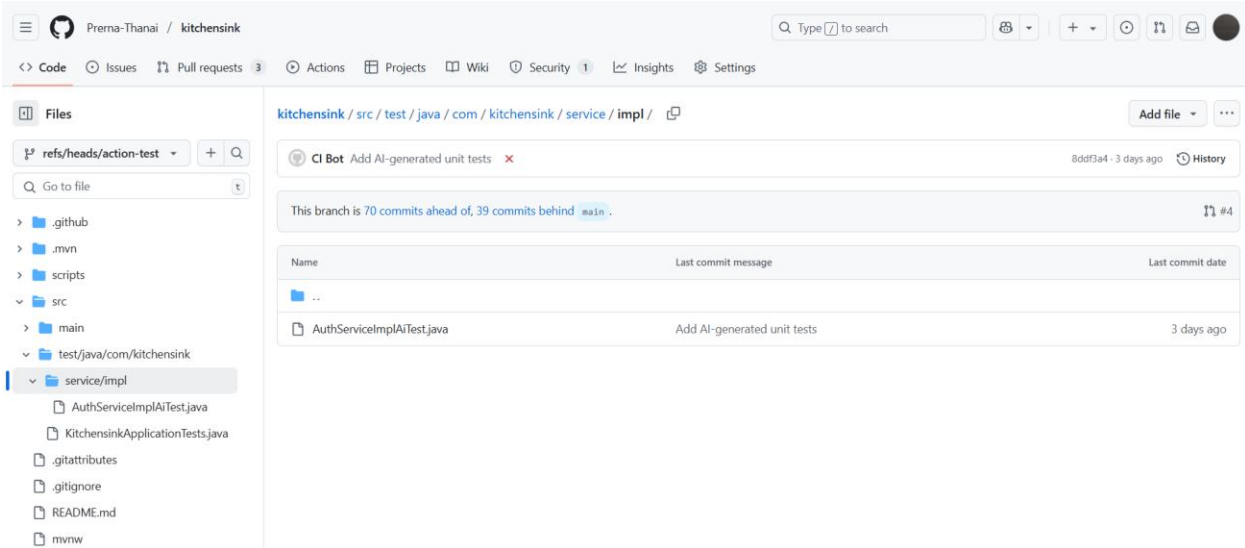
```
generate-tests
succeeded 3 days ago in 21s

Run test generator script
36 {
37   "audio_tokens": 0,
38 },
39 "completion_tokens_details": {
40   "reasoning_tokens": 0,
41   "audio_tokens": 0,
42   "accepted_prediction_tokens": 0,
43   "rejected_prediction_tokens": 0
44 },
45 },
46 "service_tier": "default",
47 "system_fingerprint": "fp_34a54ae93c"
48 }
49 Generated test saved to src/test/java/com/kitchensink/service/impl/AuthServiceImplAiTest.java

Commit generated tests
1 ▶ Run git config user.name "CI Bot"
8 [action-test 8ddf3a4] Add AI-generated unit tests
9 1 file changed, 39 insertions(+), 41 deletions(-)
10 To https://github.com/Prerna-Thandai/kitchensink
11 3c6add6d..8ddf3a4 action-test -> action-test

> Post Run actions/checkout@v4
> Complete job
```

Git commit with AI generated tests –



4. Additional Considerations

- **Security:** Migrated from custom logic to Spring Security with OAuth2 support

Contributor - Prerna