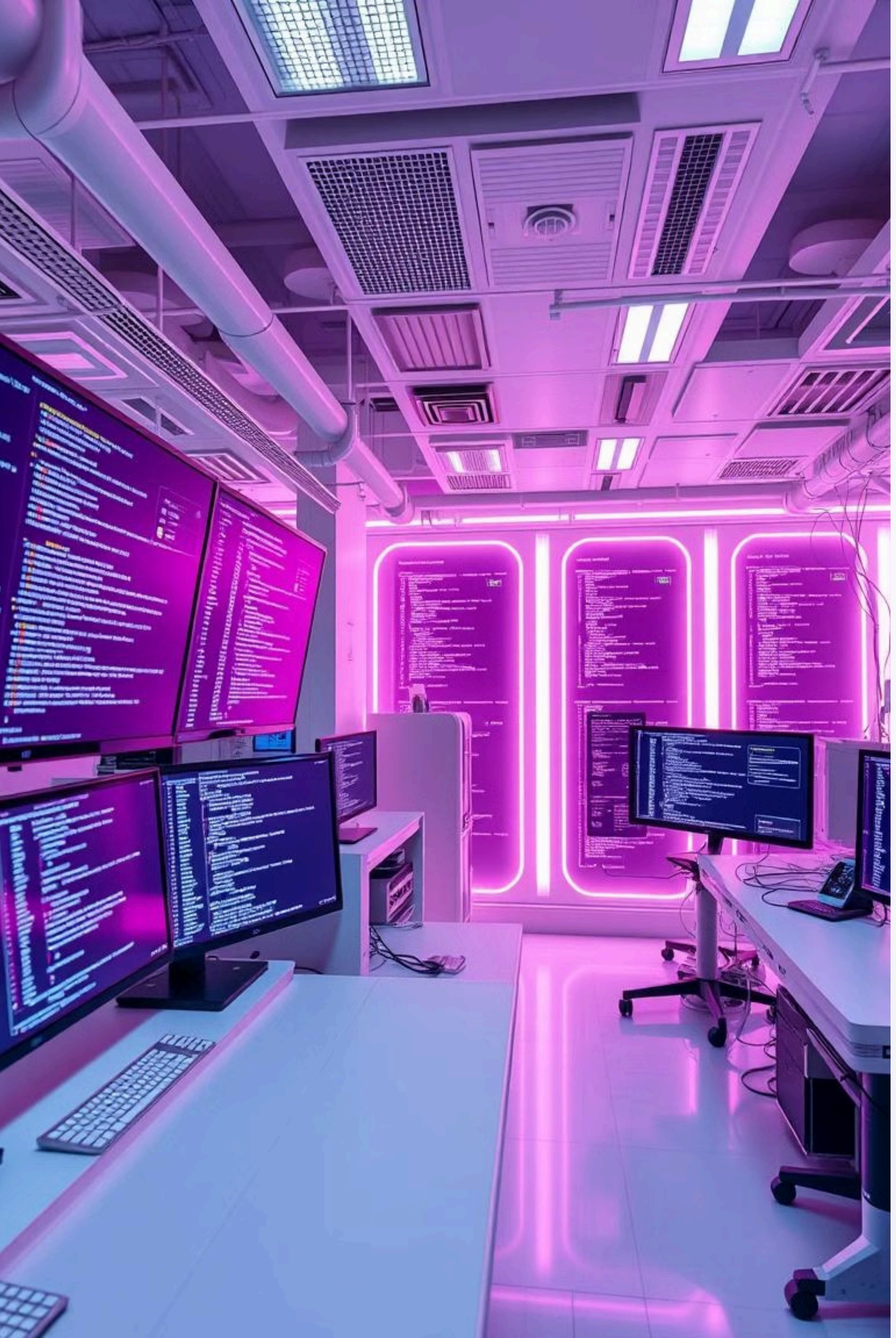
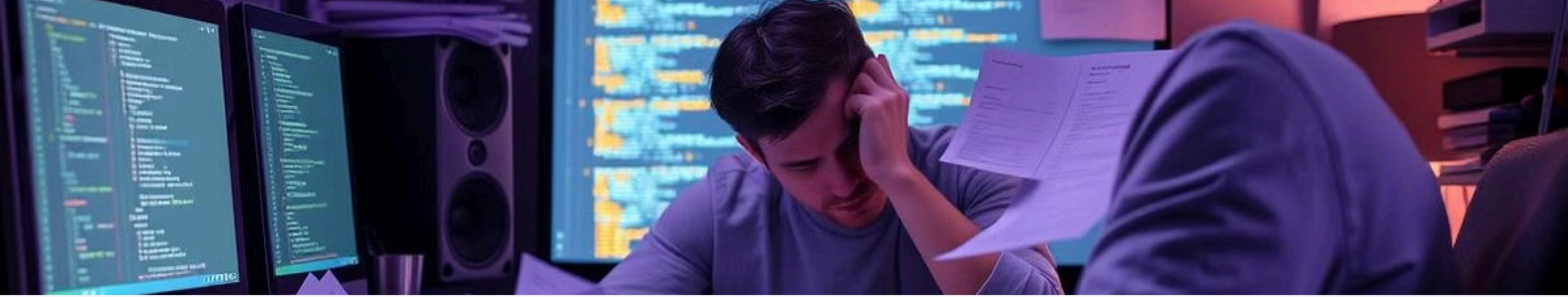


Context Aware Test Case Generation using LLM

Revolutionizing Java testing through AI-driven automation. Generating functional test cases using a Large Language Model. Two key functionalities: Full generation and Incremental updates.

Creators: Devashish, Manvi, Nishta, and Sandeepan



The Challenge: Traditional Java Testing

Manual Efforts

Manual test case creation is time-consuming and error-prone.

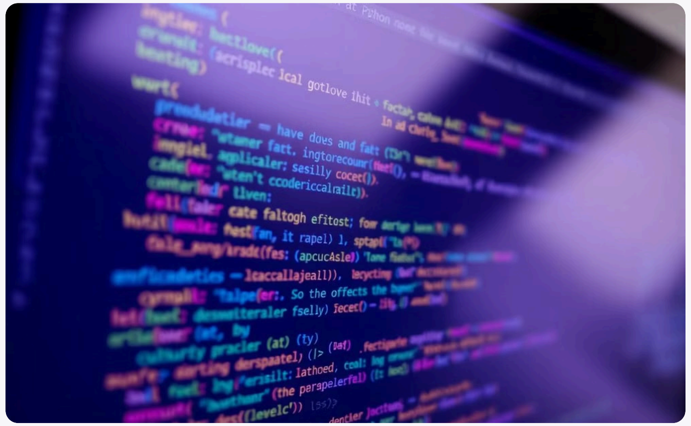
Coverage Gaps

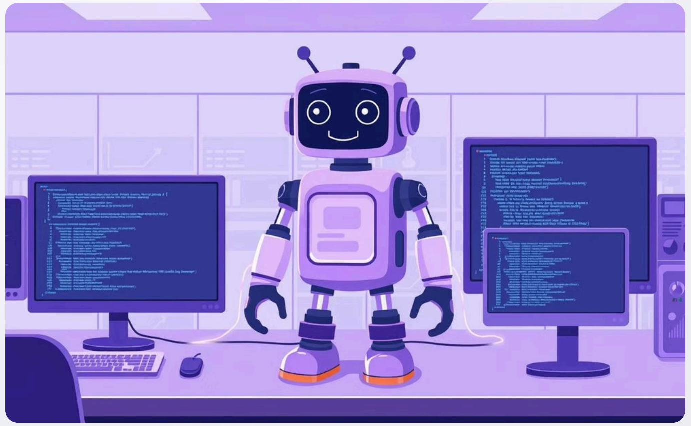
Maintaining test coverage across complex Java codebases is difficult.

Adaptability

Current tools often lack the intelligence to adapt to code changes efficiently.

Introducing the AI Solution: DeepSeek-Powered Test Generator





Tech Stack Overview

Python for model script, Java SpringBoot for test code, React for frontend, Cucumber for test cases.

Approach

AI-driven test generator leverages LLMs to parse Java codebase, map context, and extract features.

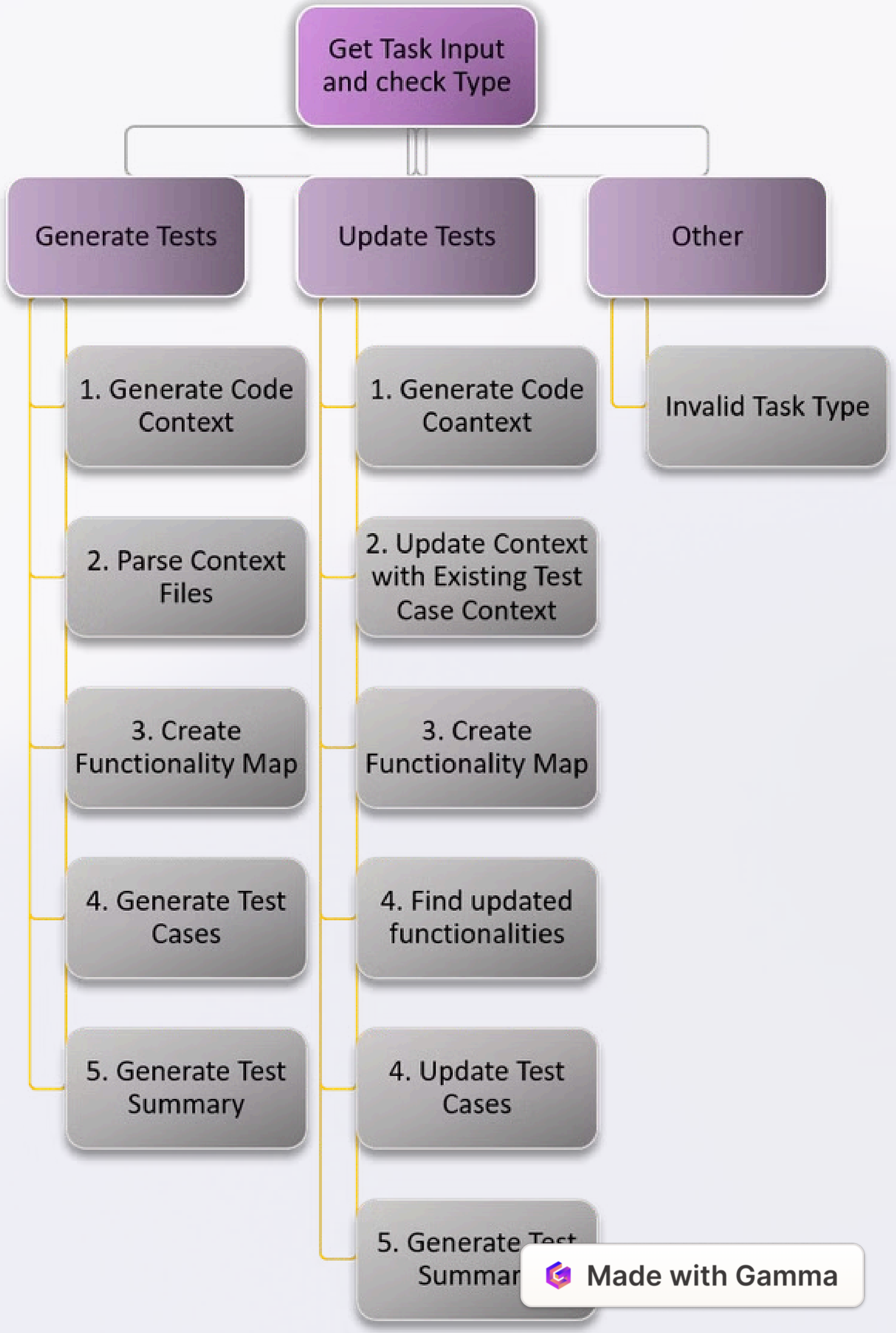
LLM Integration

Enables automatic generation of comprehensive and effective test cases, reducing manual effort and improving test coverage.

Full Test Case Generation: Comprehensive Coverage

The AI meticulously handles two primary tasks: generating new tests and updating existing tests.

1. **Generate Tests:** The system starts by creating code context. It parses files and maps functionalities.
2. **Update Tests:** The system updates existing test cases by finding updated functionalities and generating a summary.



Get Task Input and check Type
Generate Tests
Update Tests
Other
1. Generate Code Context
2. Parse Context Files
3. Create Functionality Map
4. Generate Test Cases
5. Generate Test Summary
1. Generate Code Coantext
2. Update Context with Existing Test Case Context
3. Create Functionality Map
4. Find updated functionalities
4. Update Test Cases
5. Generate Test Summar
Invalid Task Type

Made with Gamma

Benefits: Streamlining the Java Testing Process

99.79%

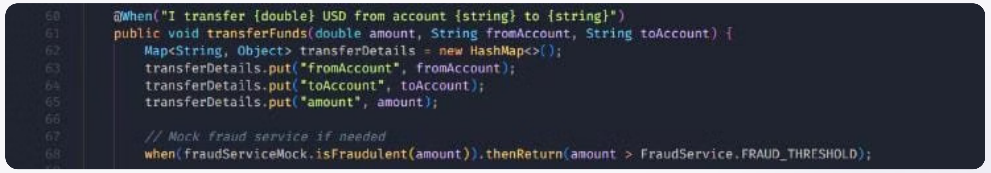
Testing Time Reduction

Reduced Time from 12 days of a developer effort to 12 mins

100%

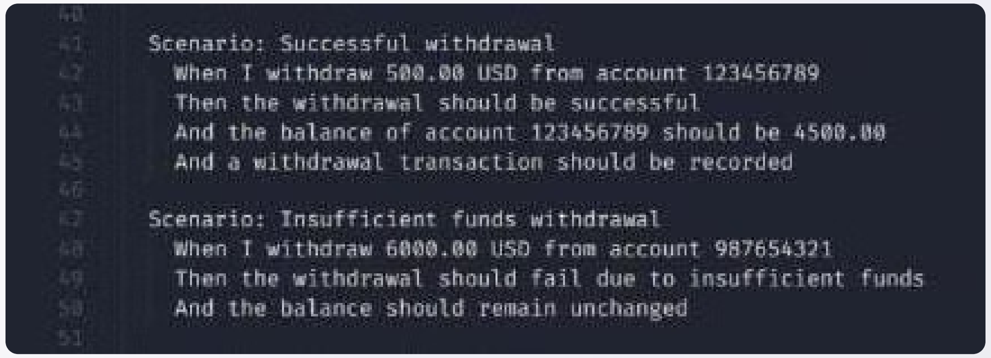
Higher Functionality Test Coverage

Covered all functionalities.



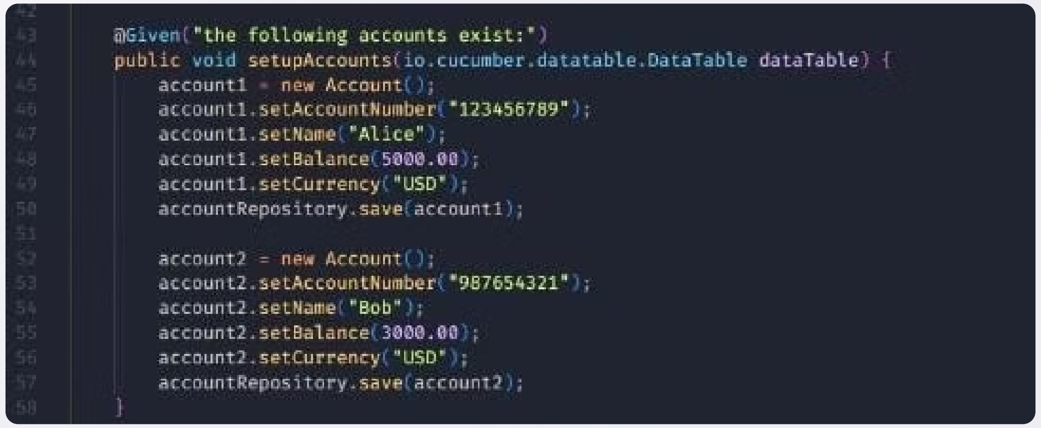
```java
@When("I transfer {double} USD from account {string} to {string}")
public void transferFunds(double amount, String fromAccount, String toAccount) {
    Map<String, Object> transferDetails = new HashMap<>();
    transferDetails.put("fromAccount", fromAccount);
    transferDetails.put("toAccount", toAccount);
    transferDetails.put("amount", amount);

    // Mock fraud service if needed
    when(fraudServiceMock.isFraudulent(amount)).thenReturn(amount > FraudService.FRAUD_THRESHOLD);
```



```gherkin
  Scenario: Successful withdrawal
    When I withdraw 500.00 USD from account 123456789
    Then the withdrawal should be successful
    And the balance of account 123456789 should be 4500.00
    And a withdrawal transaction should be recorded

  Scenario: Insufficient funds withdrawal
    When I withdraw 6000.00 USD from account 987654321
    Then the withdrawal should fail due to insufficient funds
    And the balance should remain unchanged
```



```java
@Given("the following accounts exist:")
public void setupAccounts(io.cucumber.datatable.DataTable dataTable) {
    account1 = new Account();
    account1.setAccountNumber("123456789");
    account1.setName("Alice");
    account1.setBalance(5000.00);
    account1.setCurrency("USD");
    accountRepository.save(account1);

    account2 = new Account();
    account2.setAccountNumber("987654321");
    account2.setName("Bob");
    account2.setBalance(3000.00);
    account2.setCurrency("USD");
    accountRepository.save(account2);
}
```

Made with Gamma

Future Works

VS Code Extension

Develop a VS Code extension to streamline the testing workflow.

Multiple Languages Support

Extend AI capabilities to support test case generation for other languages.

UI Test Automation

Incorporate UI test automation, enhancing end-to-end testing coverage.

Thankyou