PM2

KanBan Storyboard tasks:

Below is a set of 14 tasks that are both in progress and that still need to be worked on. Each task has its own story point value (showing how significant it is to the overall

1. Define Scope of Test Case Generator
    a. Story Points: 8
    b. Essential first step for clear project boundaries and requirements.
2. Gather Requirements for Test Case Formats
    a. Story Points: 5
    b. Understanding the format is necessary before any generation logic is built. Understanding what we want the output and inputs to look like.
3. Identify Target Application for Test Cases
    a. Story Points: 3
    b. Specifics of the application will dictate the test case parameters, creating a tech stack as well
4. Set Up Development Environment
    a. Story Points: 2
    b. Having the right tools and environment is crucial for development efficiency. Making sure we have set up the git repositories properly in order to effectively work in the future
5. Learn the ChatGPT API
    a. Story Points: 5
    b. Integral for leveraging GPT-based features in the test case generation.
6. Design Test Case Data Model
    a. Story Points: 8
    b. A well-defined data model is crucial for a scalable and robust generator. Preliminary product design should be finished
7. Creating HTML + CSS Based Frontend
    a. Story Points: 13
    b. The interface is needed for user interaction and to trigger test case generation. Must create a user interface with a usable platform
8. Create Backend Structure for API
    a. Story Points: 8
    b. The backend serves as the foundation for handling API calls.
9. Implementing Flask to Interact with Backend
    a. Story Points: 8
    b. Flask will handle the routing and server-side logic of the application. Similar to previous step, but must call function to create test cases and display on front end
10. Create & Test Backend API Calls
    a. Story Points: 13
    b. Ensuring the backend can handle requests and serve the frontend.
11. Integrate ChatGPT API with Backend Logic
    a. Story Points: 8

b. This step allows leveraging GPT-3 for generating test cases. This is the basis of our program.

12. Perform Unit Testing on Individual Components

a. Story Points: 5

b. Critical to ensure each component works as expected before integration. Product testing before sending to production

13. Conduct Integration Testing of Entire System

a. Story Points: 8

b. Validates that all components work together harmoniously. If further development is required by clients they will be worked on.

5 use cases:

Generate Test Cases: Goal is to generate test cases automatically for a given piece of code or description

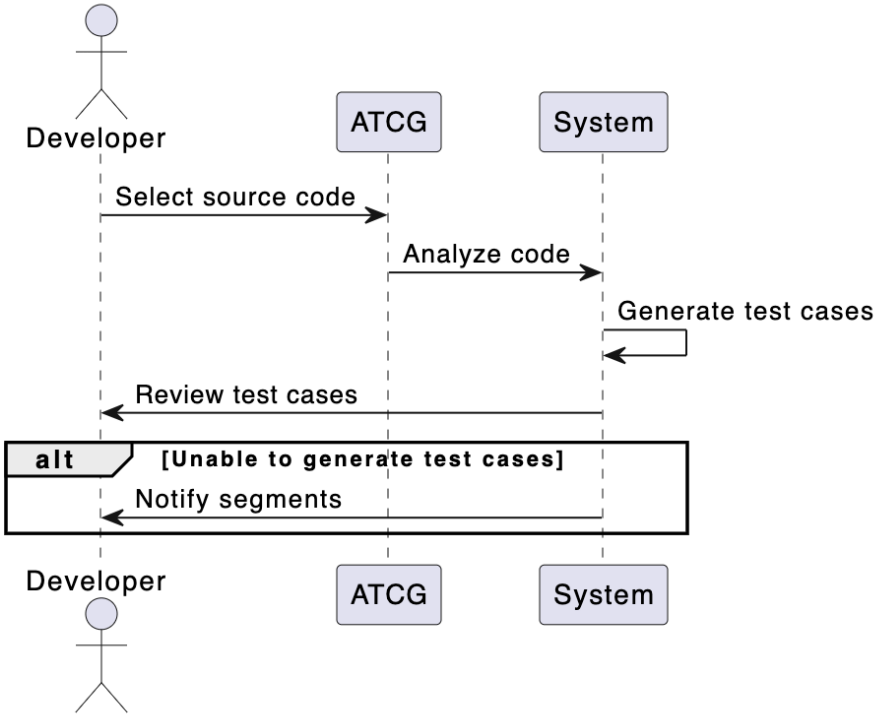
Primary Actor: Developer

Precondition: Dev already has access to source code and ATCG

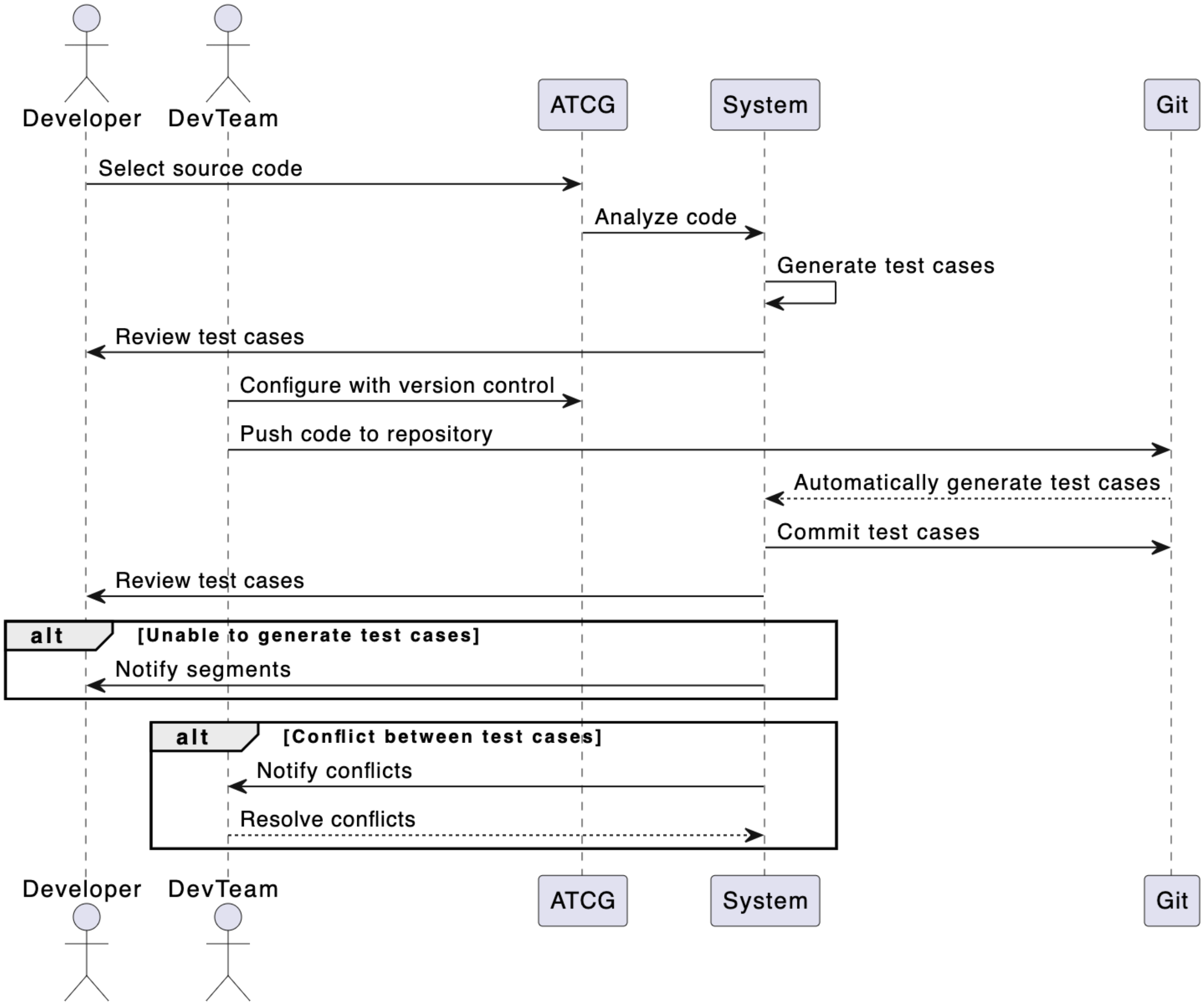
Main Scenario: Dev selects source code for with test cases are needed

Extensions: System is unable to generate test cases for specific code segments

System notifies developer of segments where it cannot be generated and developer manually completes them

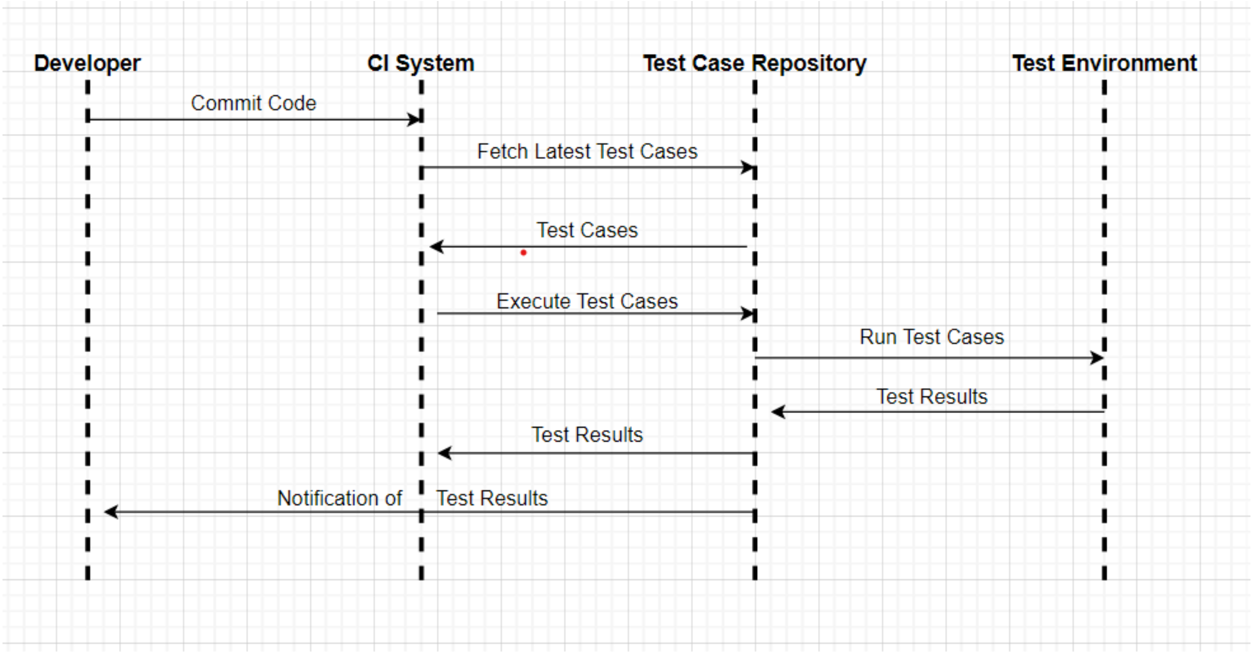


Integrate with Version Control: Goal is to integrate the Automated Test Case Generator with team's version control system
Primary Actor: Dev team
Precondition: Dev team has version control system like git
Main Scenario: Team configures ATCG w their version control system
When code is pushed to repository, sys automatically generates test cases
Extensions: Conflict between new test cases and existing ones
System notifies team of conflicts
Team resolves conflicts manually



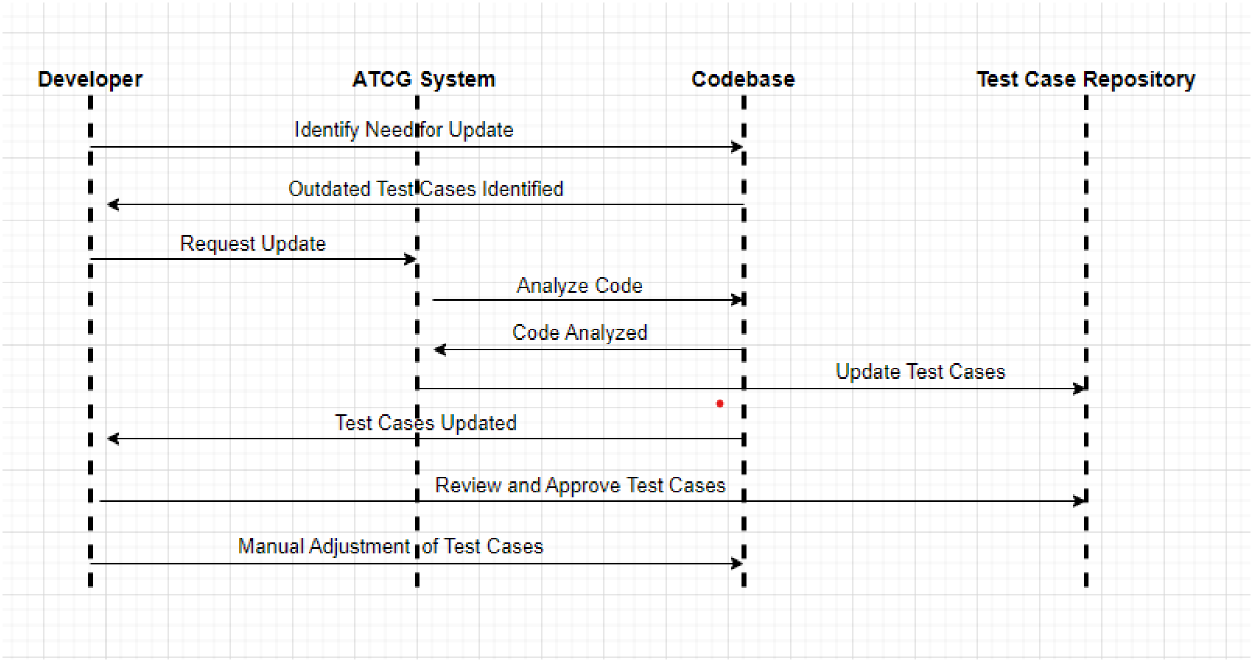
Execute Test Cases: Goal is to automatically execute generated test cases as part of the CI pipe
Primary Actor: Continuous integration system
Precondition: Test cases generated and available in repository

Main Scenario: New commit triggers CI pipeline
CI system fetches latest cases
Extensions: Some tests fail
System notifies devs who investigate failures



Update Test Cases: Goal is to update existing test cases to reflect changes in the codebase
Primary Actor: Developer
Precondition: Codebase updated and existing cases no longer valid
Main Scenario: Dev identifies outdated cases
ATCG updates test cases based on new code
Dev reviews and approves update

Extensions: Test cases do not cover code fully
Developer manually changes them

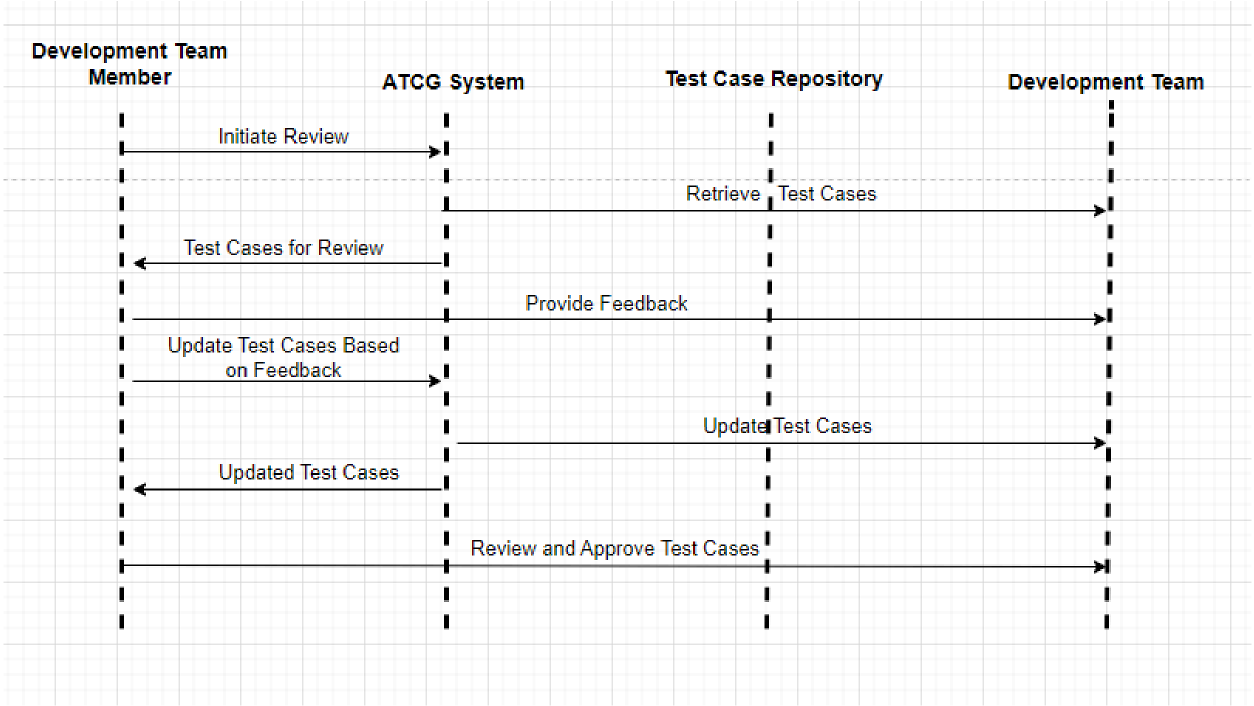


Collaborative Review of Test Cases: Goal is to collaboratively review and improve generated test cases

Primary Actor: dev team
Precondition: test cases generated and committed to repository
Main Scenario: Team member initiates review of generated test cases
Team members provide feedback and suggest improvements
Updates to test cases made based on feedback

Team approves final version

Extensions: Consensus on changes cannot be determined and dev team holds meeting to come to conclusion


Development Team Member
ATCG System
Test Case Repository
Development Team
Initiate Review
Retrieve Test Cases
Test Cases for Review
Provide Feedback
Update Test Cases Based on Feedback
Update Test Cases
Updated Test Cases
Review and Approve Test Cases