جامعة قطر
QATAR UNIVERSITY
كلية الهندسة
COLLEGE OF ENGINEERING

Spring 2025

CMPT 782 – Special Topics in Computer Science (Web Development)

Project – Phase 1

Fact-Checking for AI Chatbots: A Semi-Automated Human-in-the-Loop Approach

Submitted by:

Meshaal Al-Saffar (200607511)

Submitted to:

Dr. Abdelkarim Erradi

Date: 20 April 2025

Section 1 Introduction

1.1 Overview & Requirements

AI chatbots have become increasingly prevalent in various domains, from customer service to education and healthcare. While these chatbots offer many benefits, they also present significant challenges related to information accuracy. What if an AI provides incorrect medical advice? How can we ensure users receive trustworthy information when interacting with AI systems? Furthermore, some organizations may lack the technical expertise to validate AI outputs, while others may deploy systems without proper verification mechanisms, leading to potential misinformation.

To address the challenge of AI trustworthiness and information accuracy, this project aims to develop a semi-automated human-in-the-loop fact-checking tool that helps to ensure AI chatbots provide accurate and reliable information. This tool compares AI-generated responses with truthful responses, enabling automatic validation of straightforward facts while flagging the responses for additional human review. The tool creates a verification layer between AI systems and end-users, reducing the risk of misinformation.

1.2 Requirements

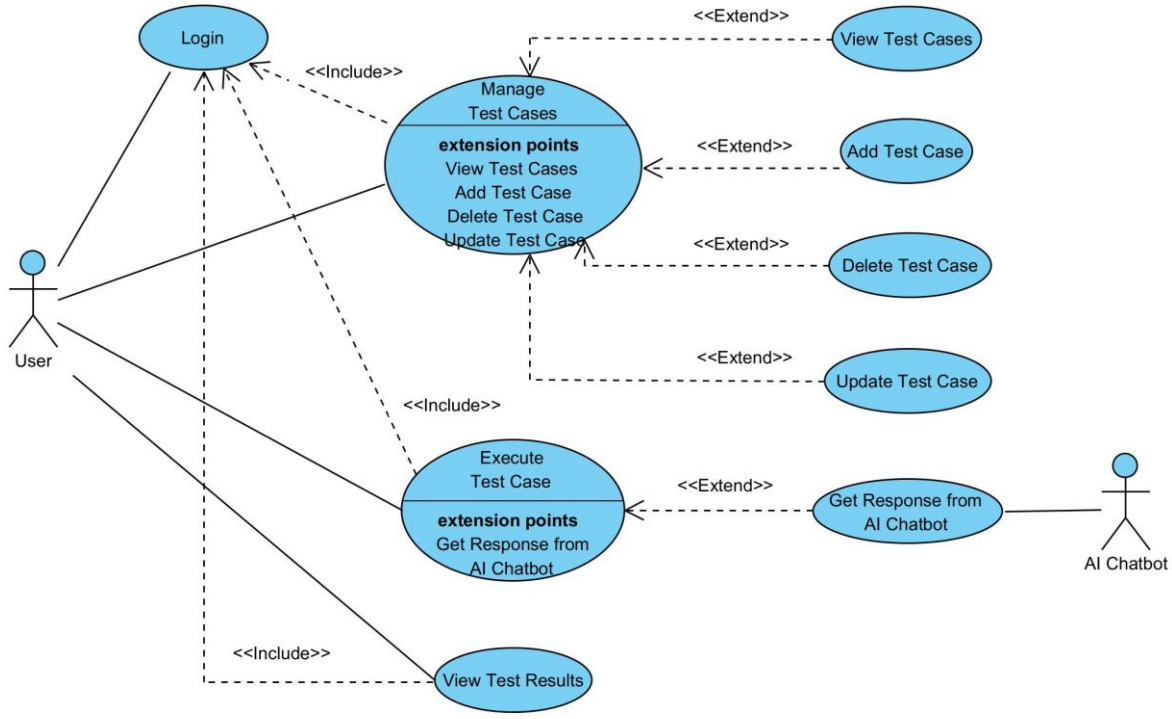


Figure: Use Cases Diagram

Table: Use Cases Description

Use Case	Description
Login	Allows users to authenticate and access the system.
Manage Test Cases	It is the generic encapsulation of the CRUD operations for the test cases (View, Add, Delete, Update).
View Test Cases	Allows users to browse through the existing test cases in the system. This helps users identify specific test cases for review or modification.
Add Test Case	Enables users to create new test cases for validating chatbot responses. When adding a test case, users must specify: - Input prompt query to be sent to the AI Chatbot - Expected "truth" response ("truth"). - Additional notes
Delete Test Case	Allows users to remove existing test cases that are no longer needed from the system. Before deletion, the system prompts for confirmation to prevent accidental removal of important test cases.
Update Test Case	Enables users to modify existing test cases to update information. Users can edit the following: - Input prompt query to be sent to the AI Chatbot - Expected "truth" response ("truth"). - Additional notes Users will not be able to edit the AI response nor the comparison analysis.
Execute Test Case	Executes the automated fact-checking validation workflow that compares chatbot responses with trusted information. The system first triggers the chatbot to get its response. Then, it compares the AI responses against trusted information provided. The comparison analysis is then saved back to the system.
Get Response from AI Chatbot	Captures responses generated by AI chatbots. The system saves the responses to be submitted later for fact-checking.
View Test Results	Allows users to view the test cases with their fact-checking outcome that includes the AI response and the comparison analysis.

Section 2 Application Design

2.1 Entities Class Diagram

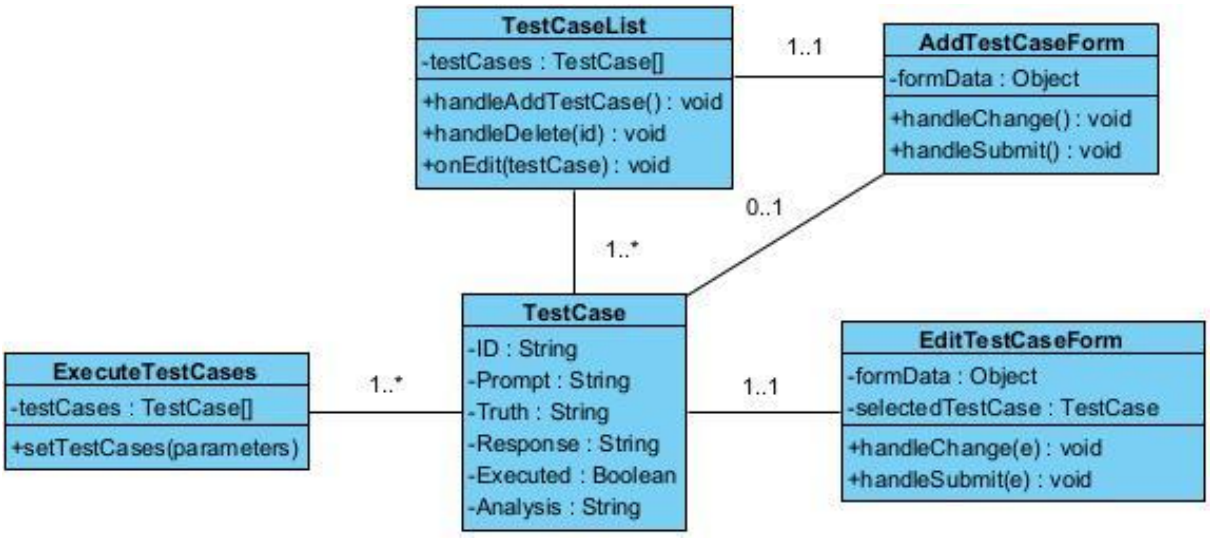


Figure: Class Diagram for Entities

TestCass

- Attributes:
    - `ID`: A unique identifier for the test case.
    - `Prompt`: The input or question for the test case.
    - `Truth`: The expected correct answer.
    - `Response`: The actual response generated or provided.
    - `Executed`: A boolean indicating whether the test case has been executed.
    - `Analysis`: A string containing the analysis or evaluation of the test case's result.

TestCaseList

- Attributes:
    - `testCases`: An array of `TestCase` objects.
- Functions:
    - `handleAddTestCase()`: Adds a new test case.
    - `handleDelete(id)`: Deletes a test case by its `ID`.
    - `onEdit(testCase)`: Edits a selected test case.

AddTestCaseForm

- Attributes:
  - `formData`: An object storing the input values for creating a new test case.
- Functions:
  - `handleChange()`: Updates the form data when the user modifies input fields.
  - `handleSubmit()`: Submits the form to create a new test case.

EditTestCaseForm

- Attributes:
  - `formData`: An object storing the updated values for the test case.
  - `selectedTestCase`: The `TestCase` being edited.
- Functions:
  - `handleChange()`: Updates the form data when the user modifies input fields.
  - `handleSubmit()`: Submits the form to update the test case.

ExecuteTestCases

- Attributes:
  - `testCases`: An array of `TestCase` objects to be executed.
- Functions:
  - `setTestCases(parameters)`: Updates the state of test cases after execution.

2.2 Entities Repositories

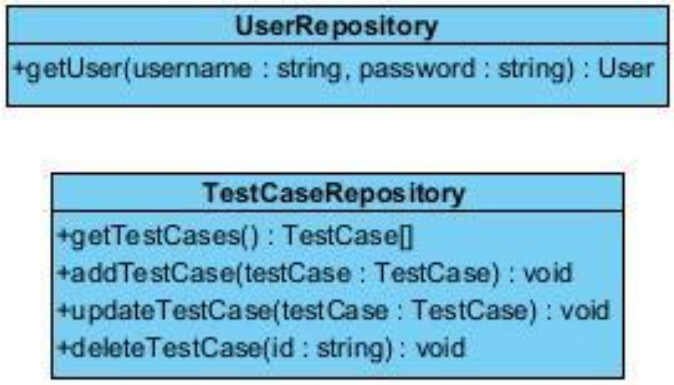


Figure: Class Diagram for Repositories

2.3 Server-Side Actions / API

Table: API Endpoints

Method	URL	Description
GET	/api/testCases	Retrieves all test cases from the data store.
POST	/api/testCases	Adds a new test case to the data store.
PATCH	/api/testCases	Updates one or more test cases in the data store.
DELETE	/api/testCases	Deletes a specific test case by its ID from the data store.
POST	/api/auth	Authenticates a user by verifying their username and password.

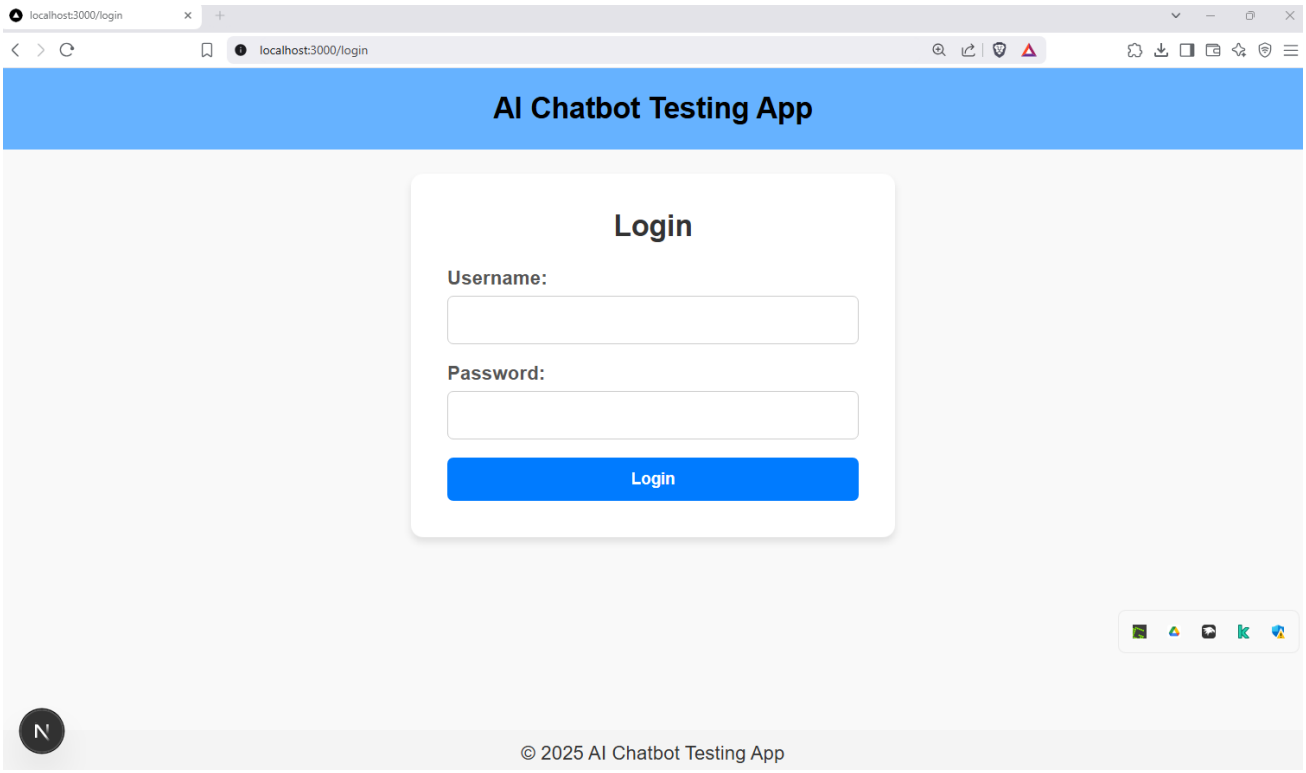
Important note about the PATCH method of the /api/testCases handler, it handles two cases:

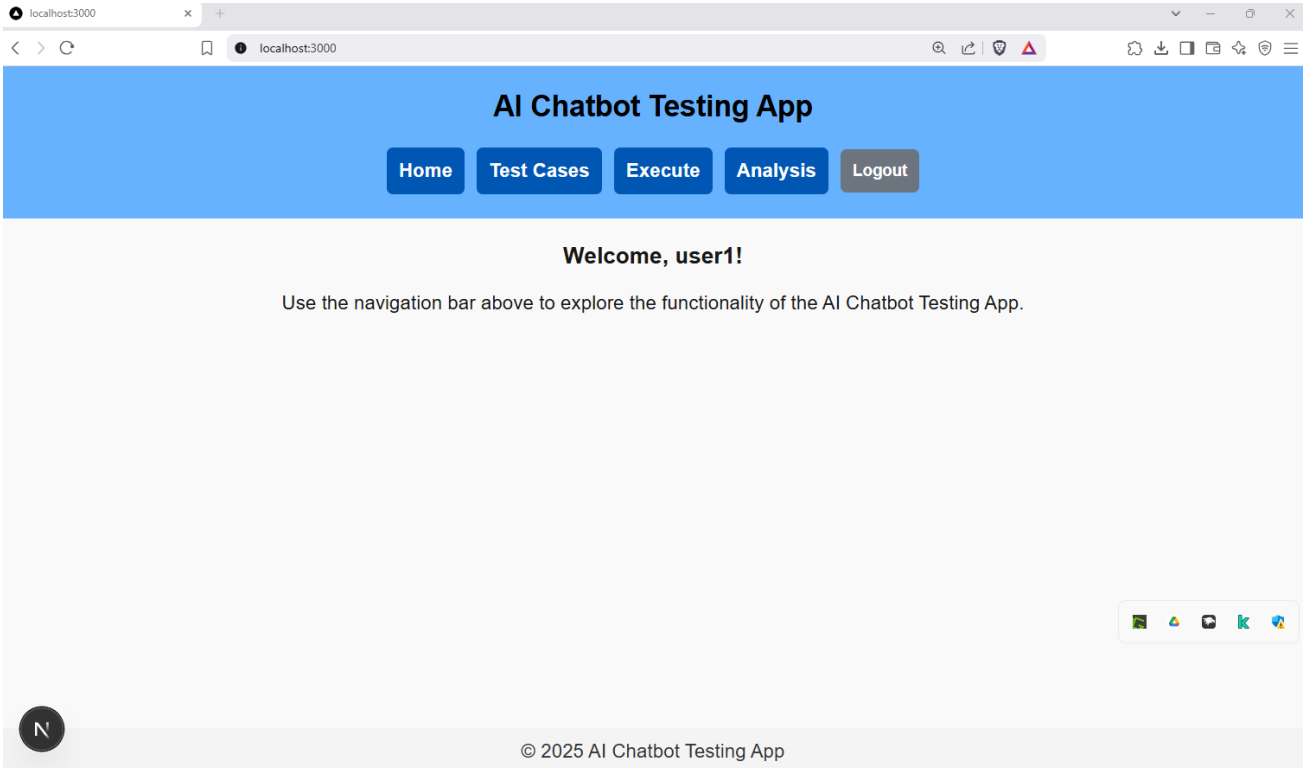
1. **Updating Multiple Test Cases (Batch Execution):**
    - If the request body is an array, it assumes the client is sending multiple test cases to be updated (e.g., for batch execution).
    - It iterates through the array of test cases, finds each test case in the data store by its ID, and updates the corresponding fields.
    - After updating all test cases, it writes the updated data back to the file.

2. **Updating a Single Test Case (Edit Operation):**
    - If the request body is not an array, it assumes the client is sending a single test case to be updated.
    - It finds the test case in the data store by its ID and updates its fields.
    - After updating the test case, it writes the updated data back to the file.

Section 3 Testing

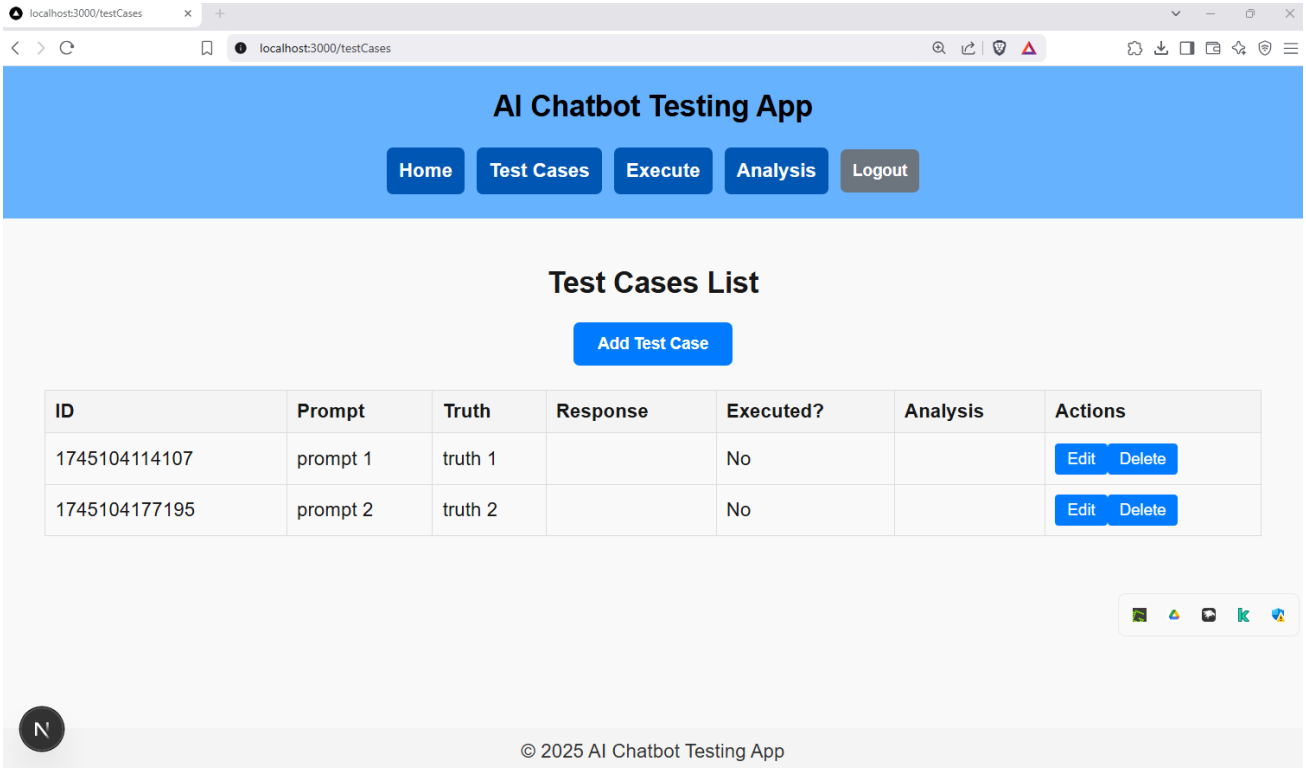
3.1 Login





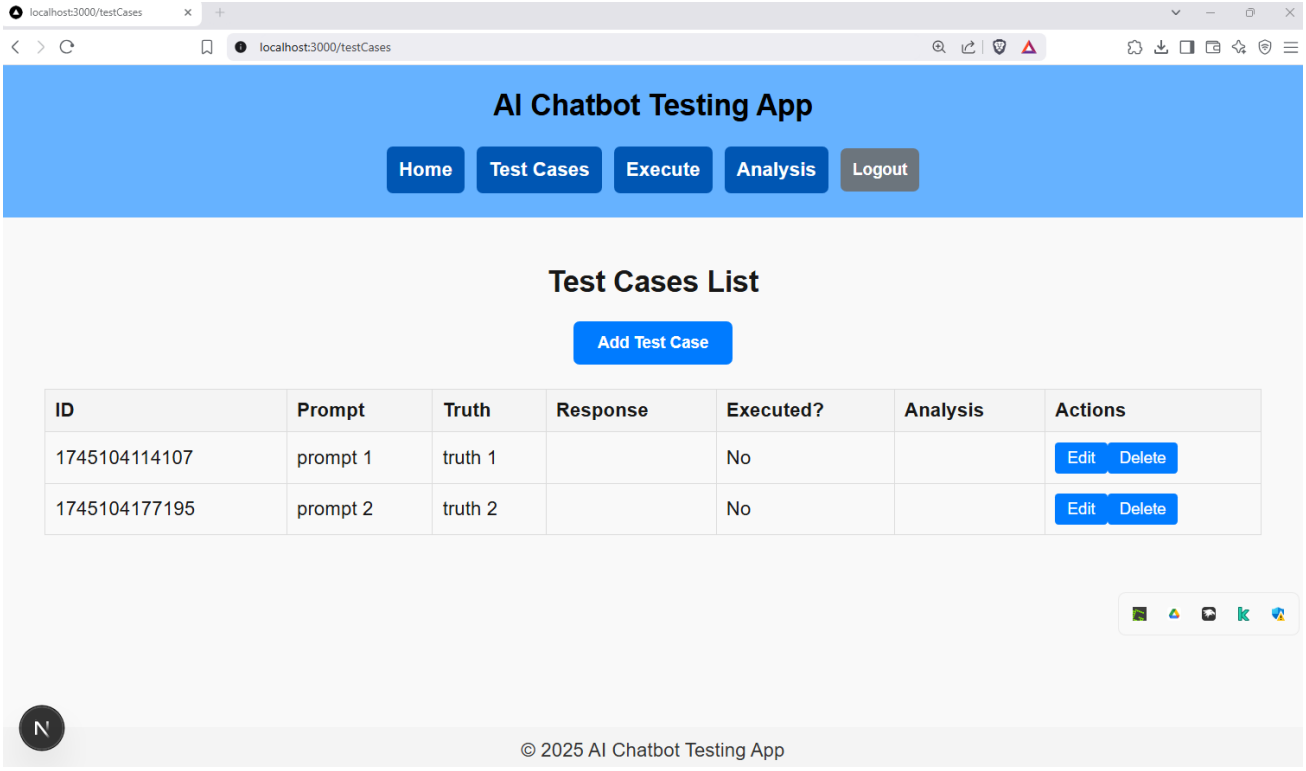
3.2 Manage Test Cases

It also covers View Test Cases

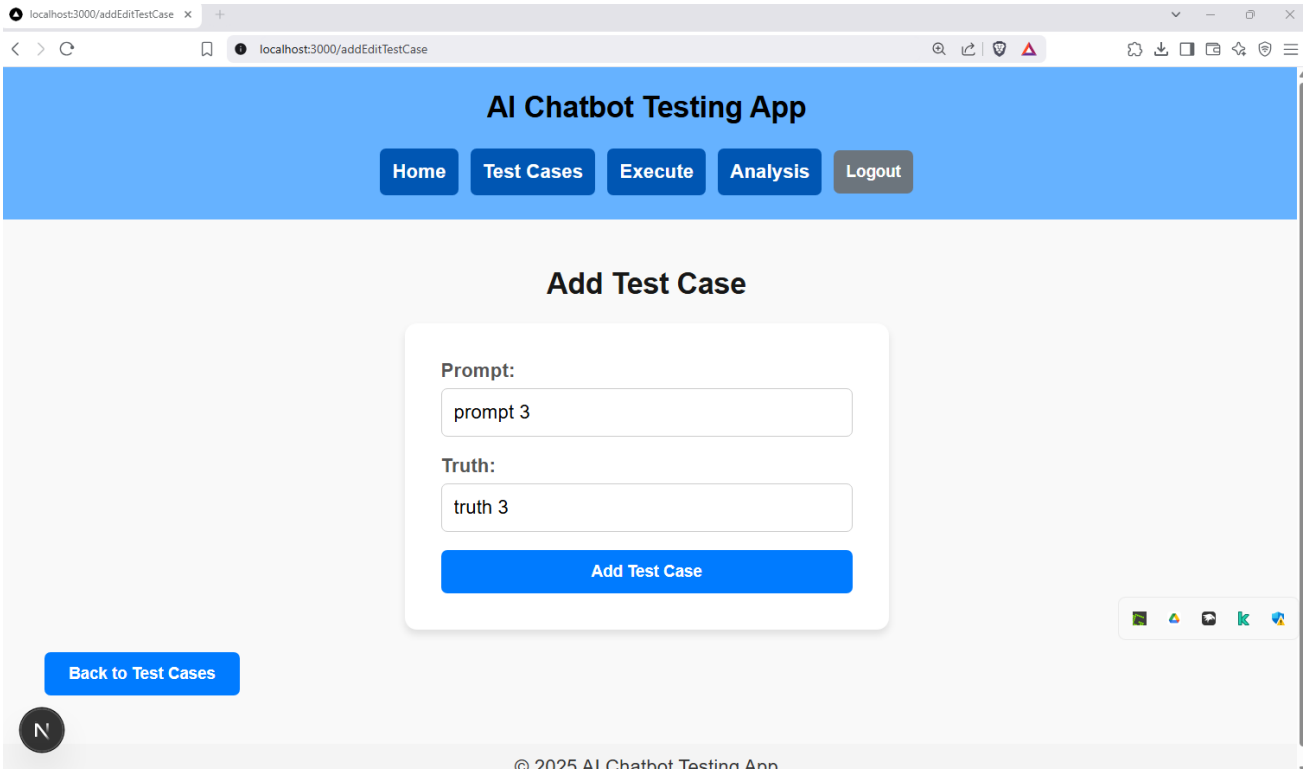


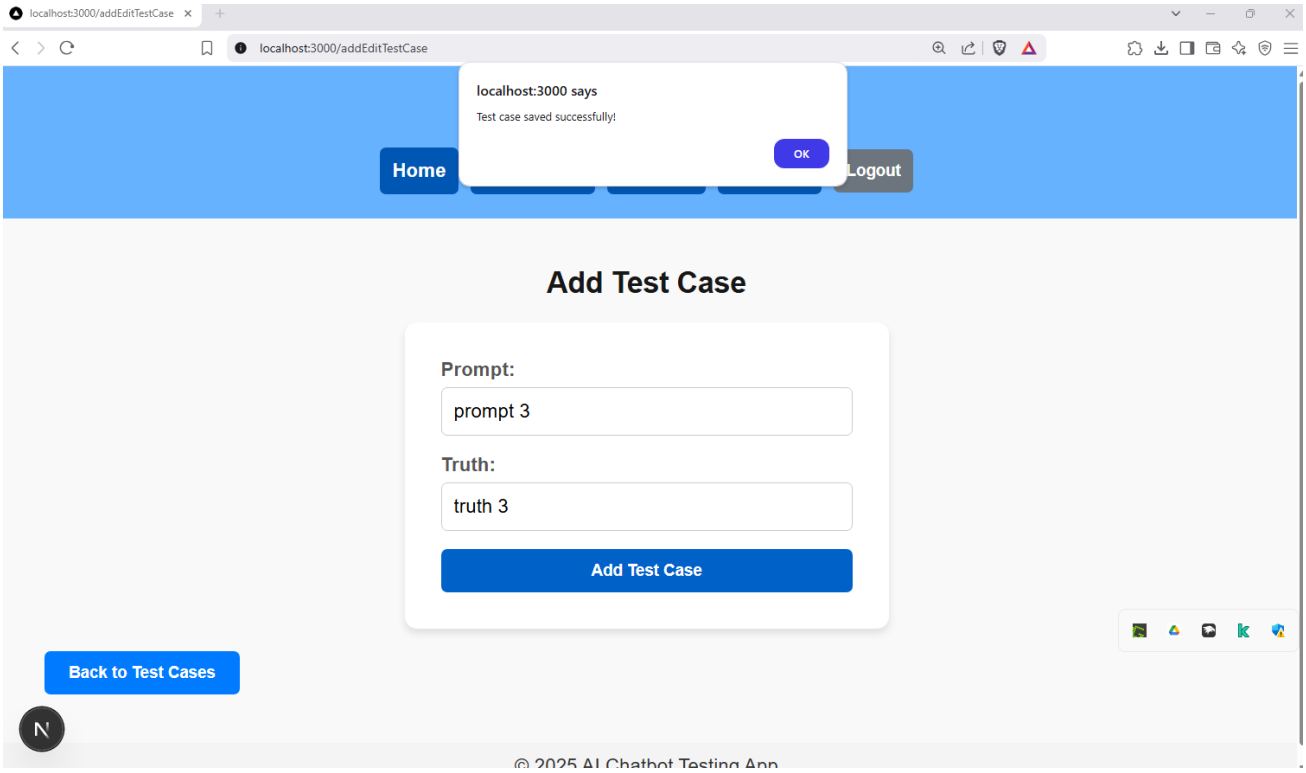
3.3 View Test Cases

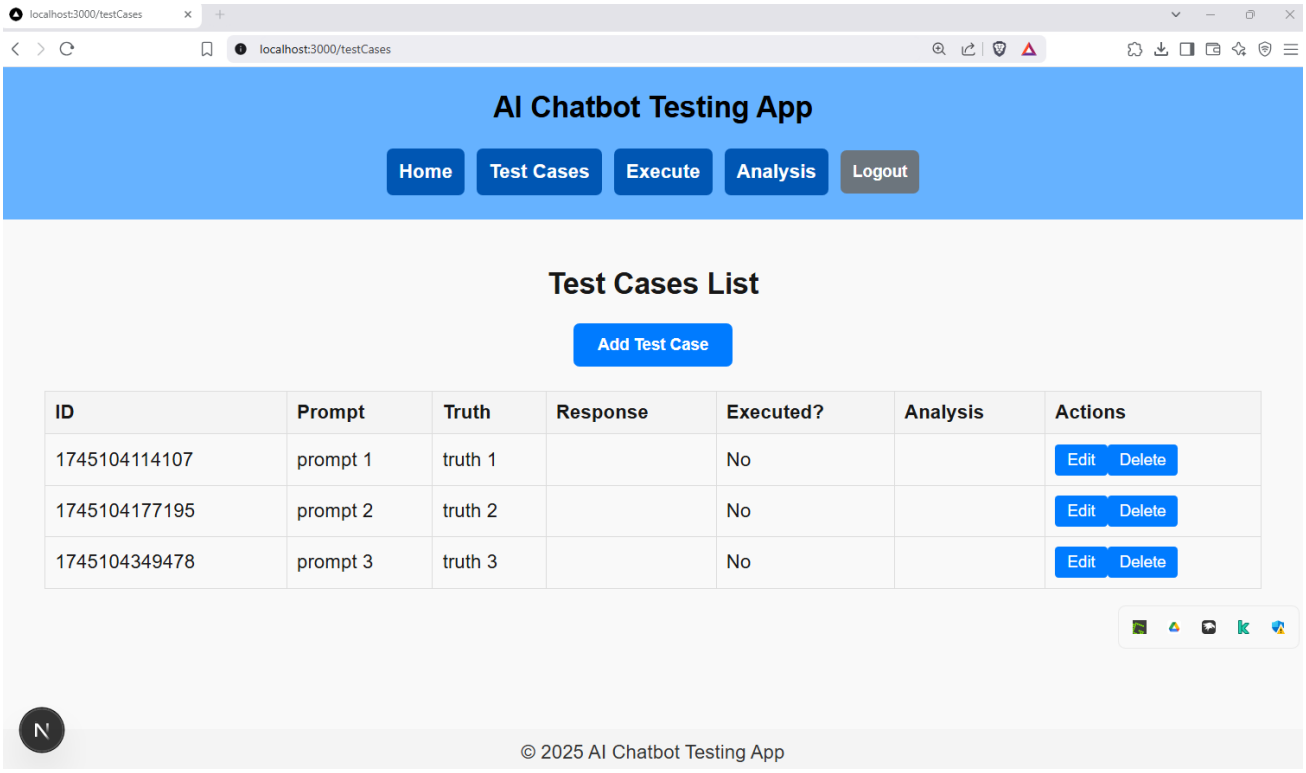
It is covered in Manage Test Cases



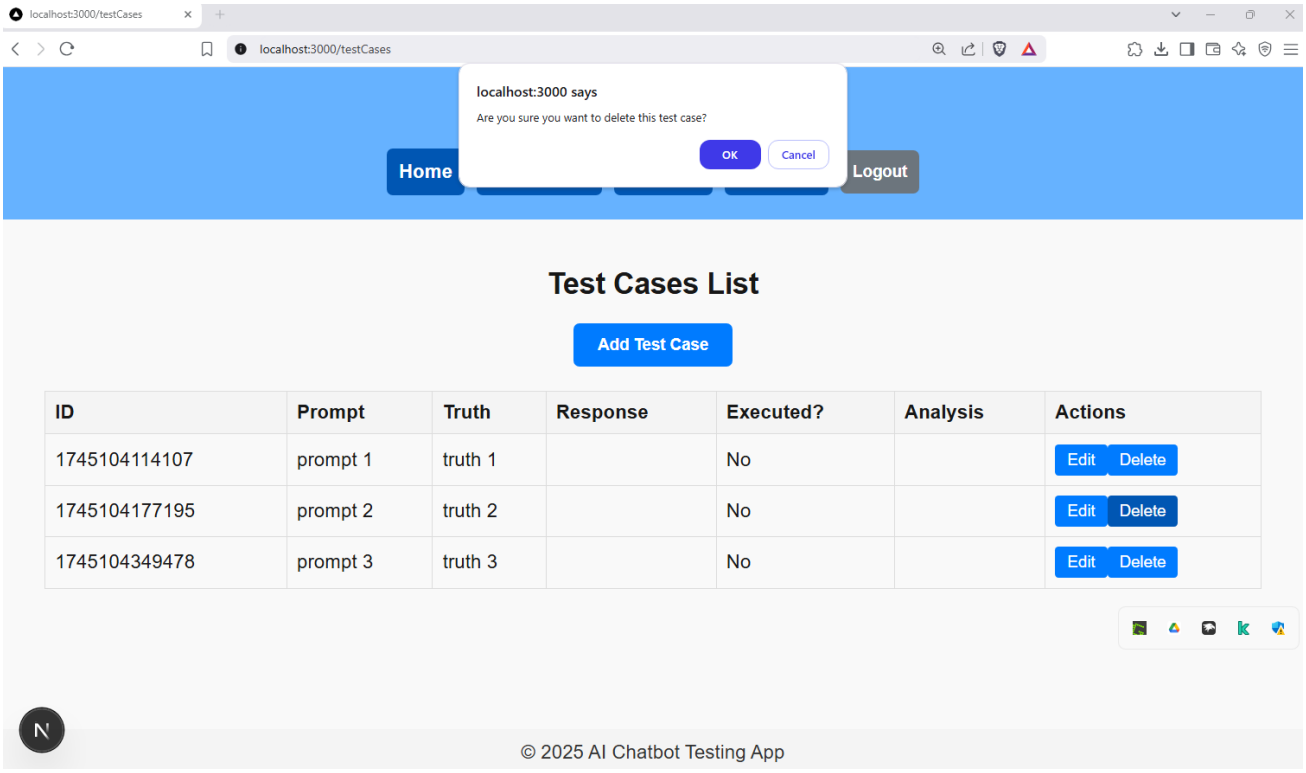
3.4 Add Test Case

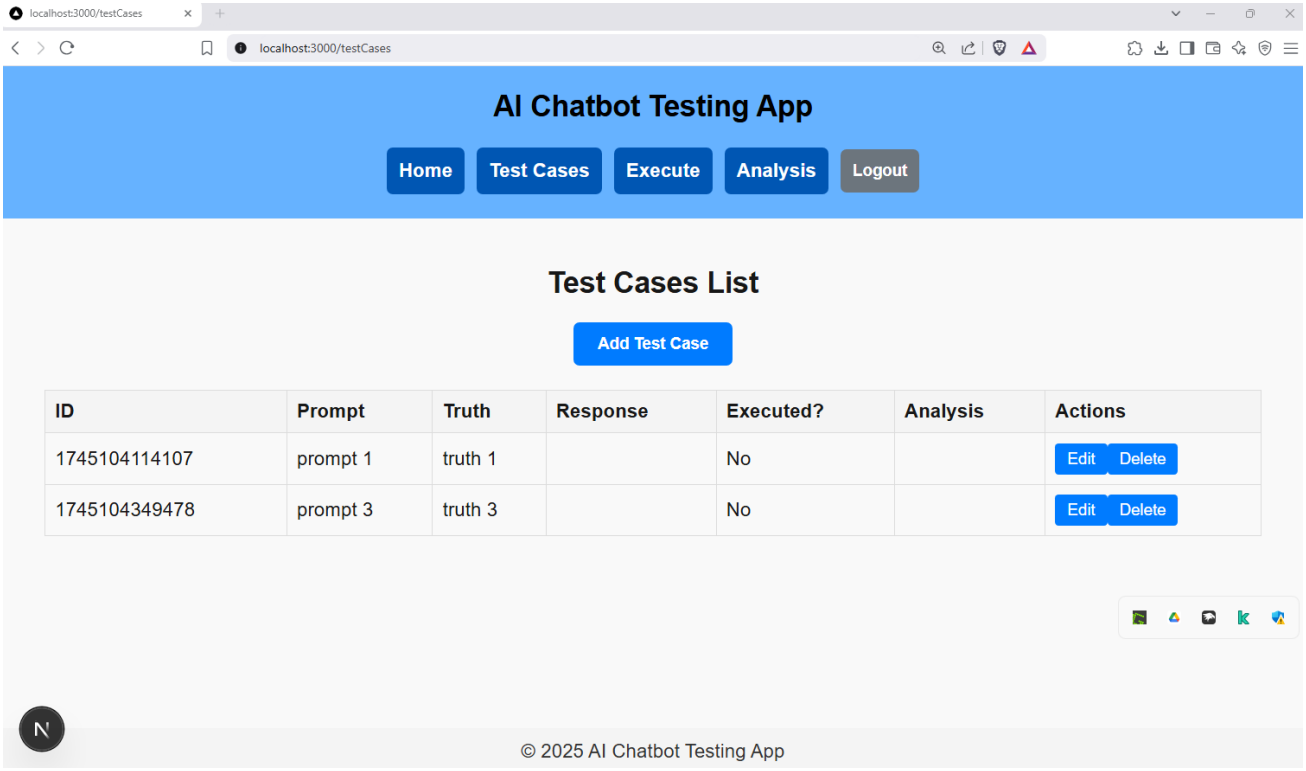




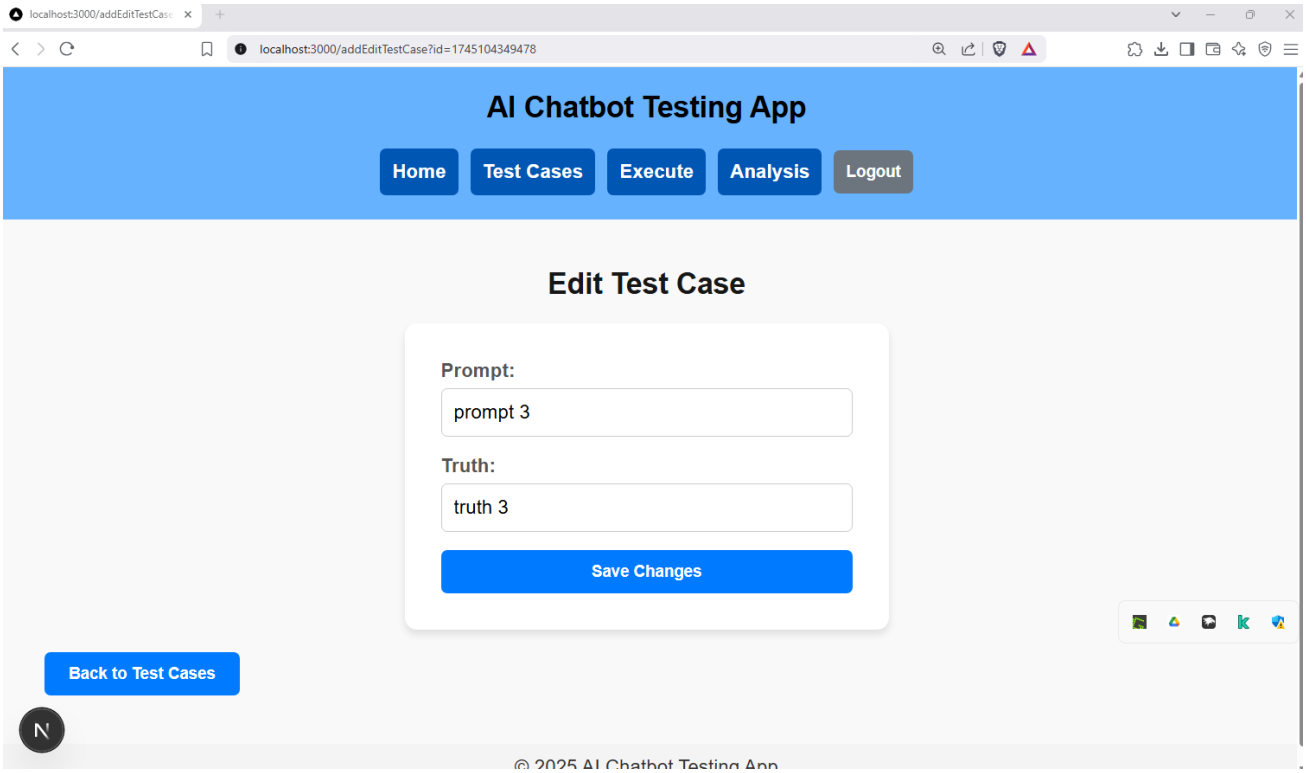

localhost:3000/testCases
localhost:3000/testCases
AI Chatbot Testing App
Home
Test Cases
Execute
Analysis
Logout
Test Cases List
Add Test Case
ID
Prompt
Truth
Response
Executed?
Analysis
Actions
1745104114107
prompt 1
truth 1
No
Edit
Delete
1745104177195
prompt 2
truth 2
No
Edit
Delete
1745104349478
prompt 3
truth 3
No
Edit
Delete
© 2025 AI Chatbot Testing App

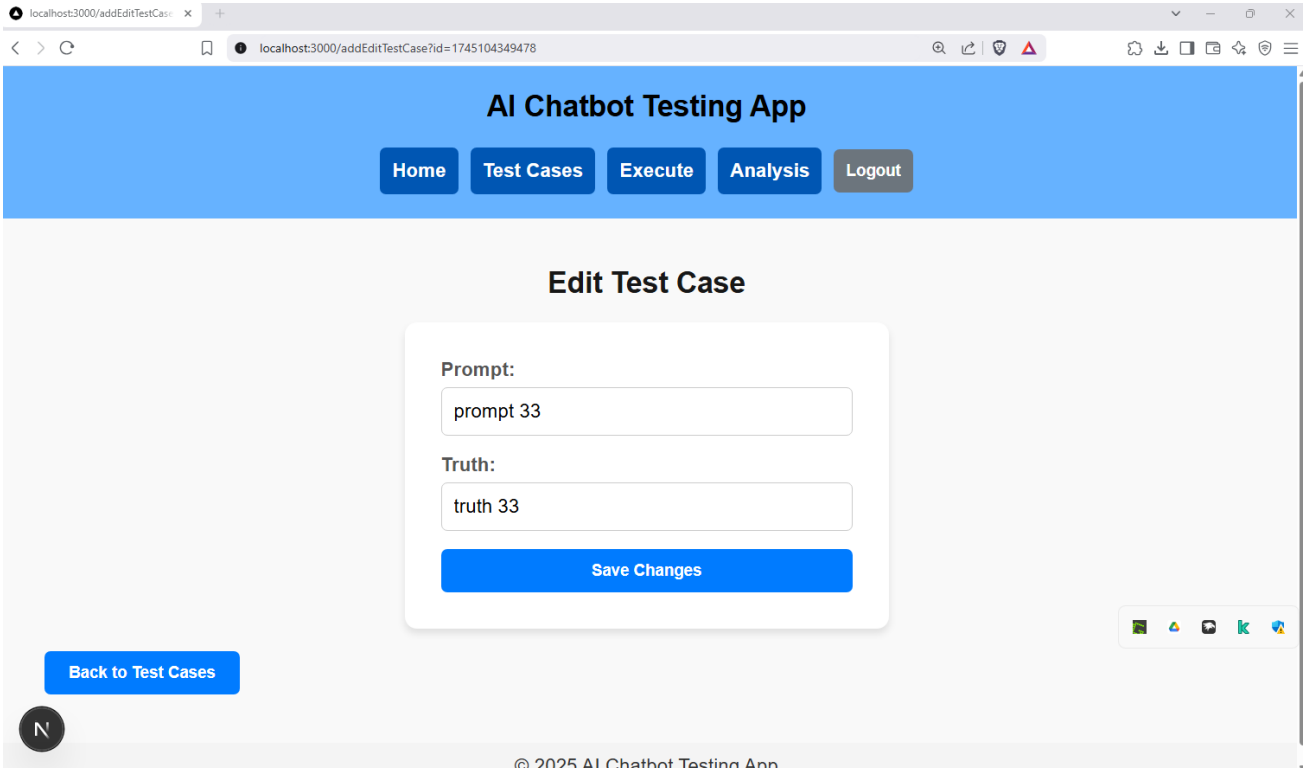
3.5 Delete Test Case

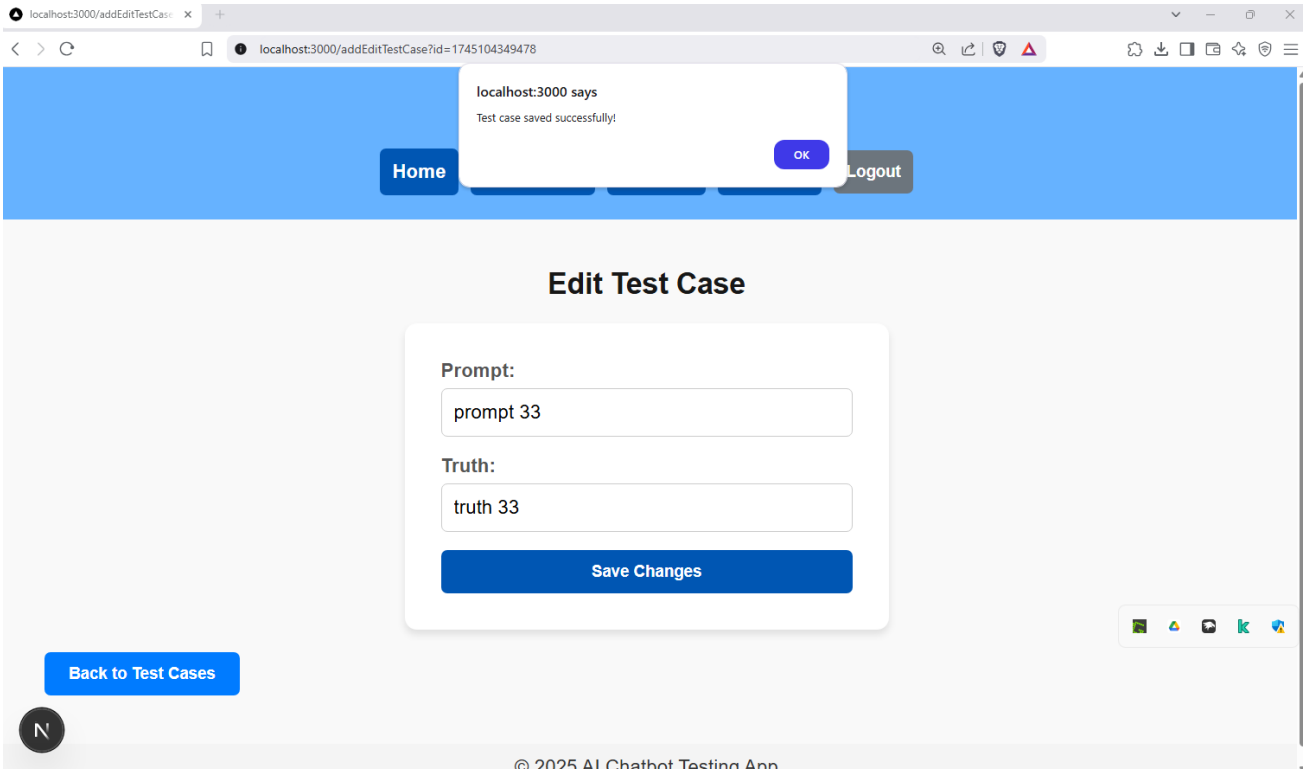


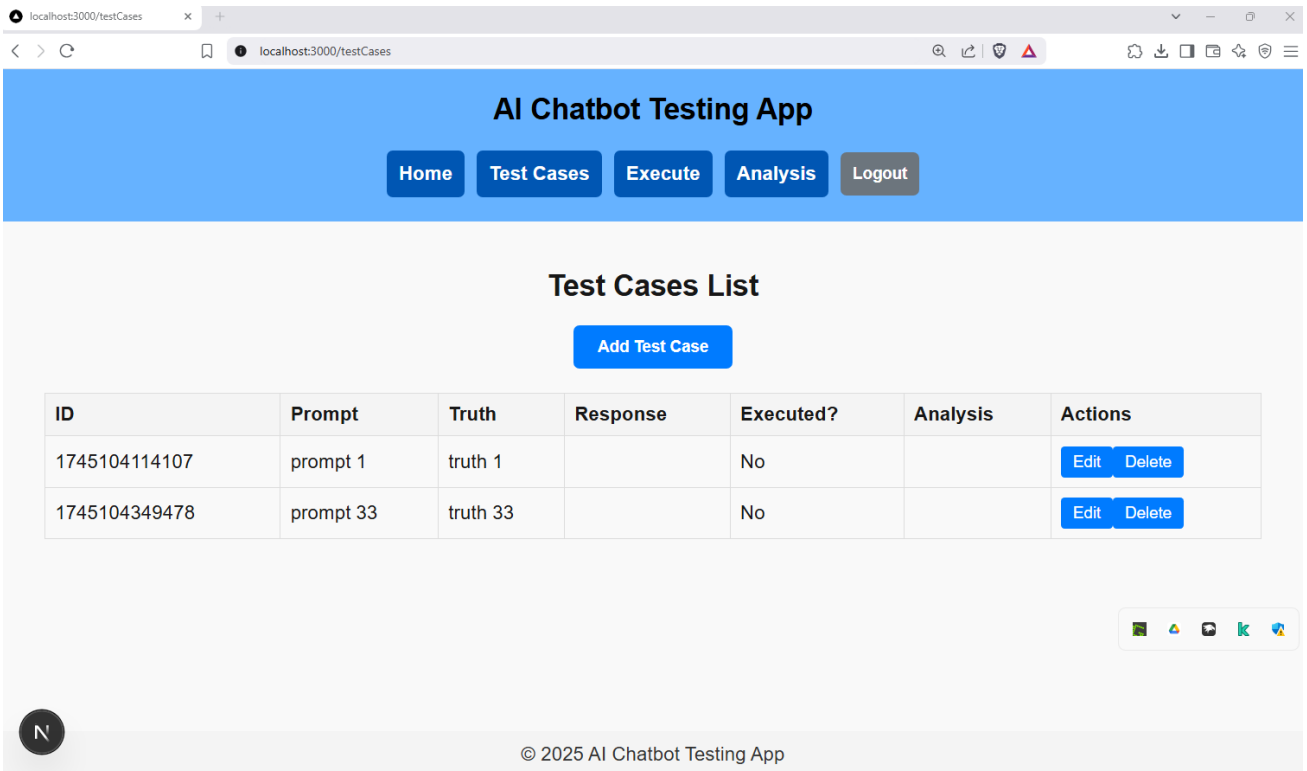


3.6 Update Test Case

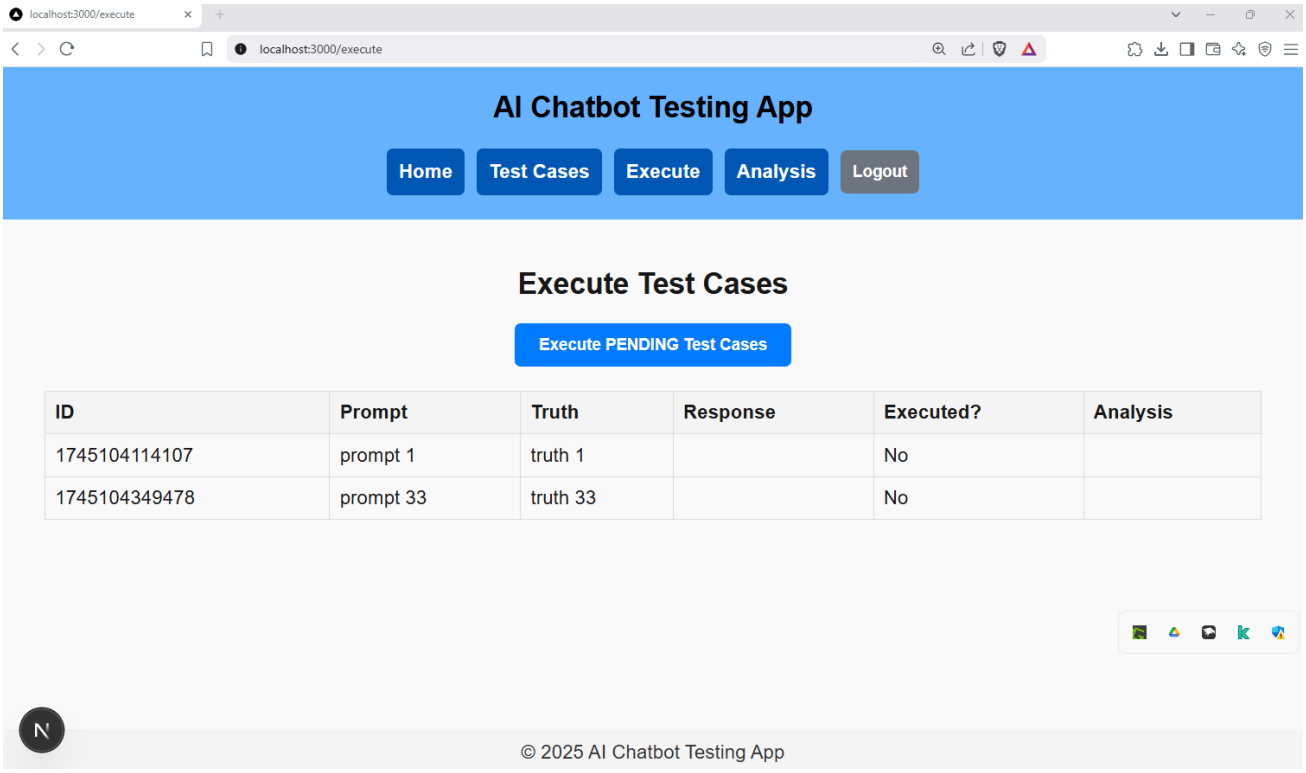


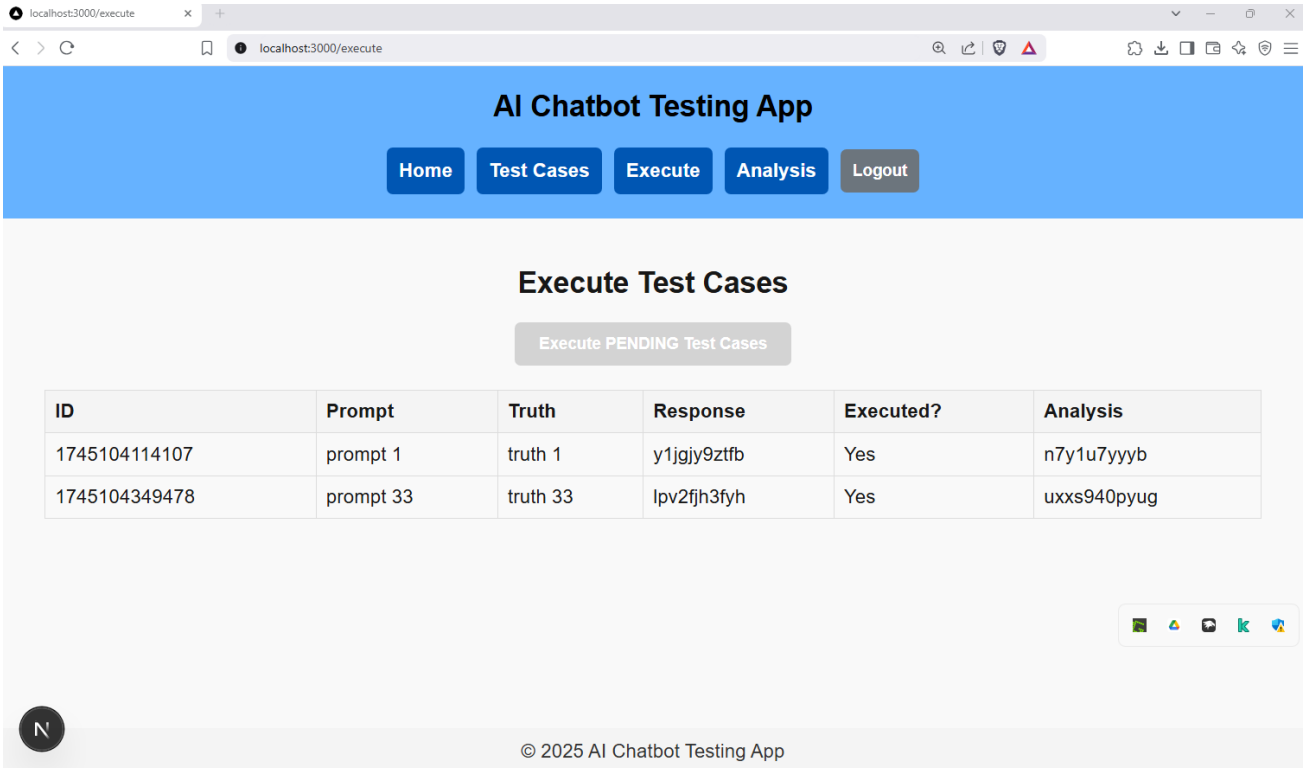



localhost:3000/addEditTestCase?id=1745104349478
localhost:3000 says
Test case saved successfully!
OK
Home
Logout
Edit Test Case
Prompt:
prompt 33
Truth:
truth 33
Save Changes
Back to Test Cases
© 2025 AI Chatbot Testing App


localhost:3000/testCases
AI Chatbot Testing App
Home
Test Cases
Execute
Analysis
Logout
Test Cases List
Add Test Case
ID
Prompt
Truth
Response
Executed?
Analysis
Actions
1745104114107
prompt 1
truth 1
No
Edit
Delete
1745104349478
prompt 33
truth 33
No
Edit
Delete
© 2025 AI Chatbot Testing App

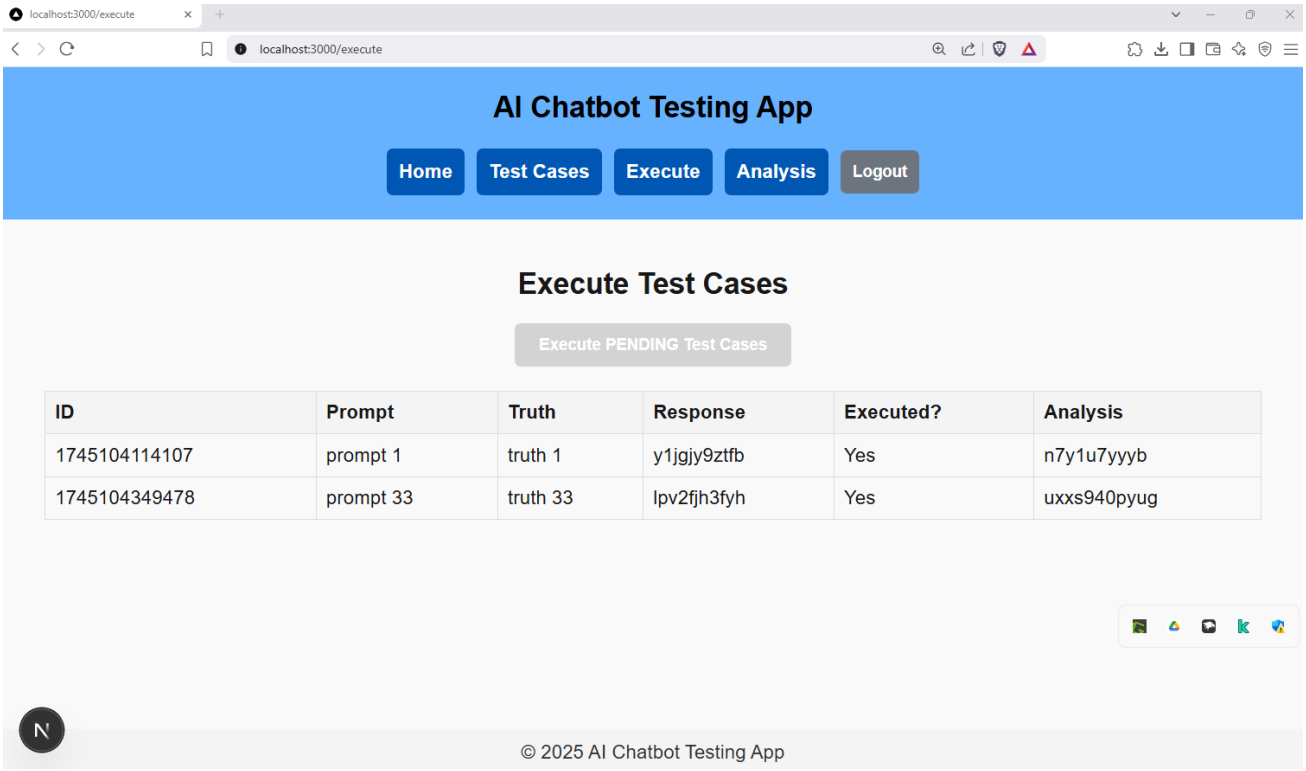
3.7 Execute Test Case



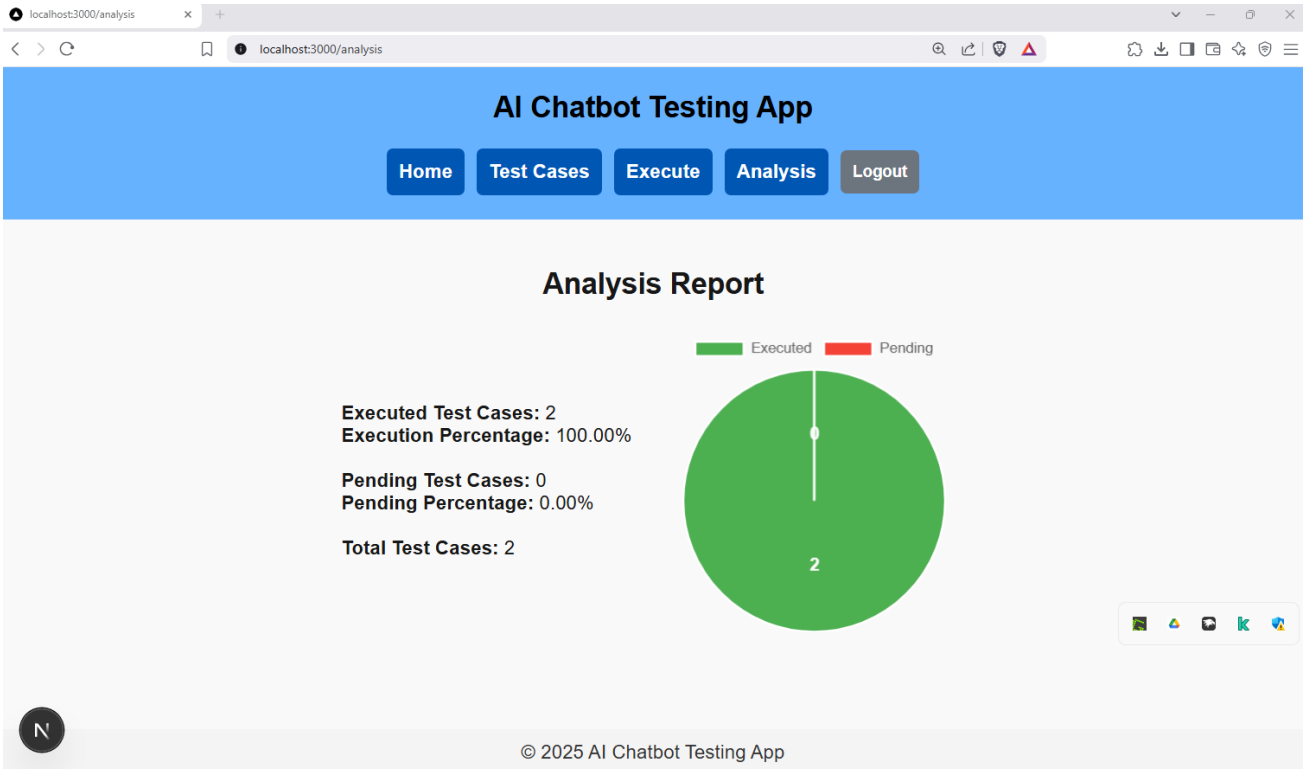


3.8 Get Response from AI Chatbot

It is covered in Execute Test Case



3.9 View Test Results



If we add another test case, it will show as one pending execution: