BLACK BOX TESTING REPORT

Assignment 2

SUMMIYA HANIA JAFRI (22FA-034-SE)
& AREESHA FEROZ (22FA-057-SE)

CHAPTER: TESTING METHODOLOGY

Introduction

Testing forms a key step in modern software development and serves as a corroborating step to demonstrate that a newly created system meets the requirements and can be efficiently used in a variety of conditions. In this chapter, the test strategy applied in order to test the proposed AI-based Health and Fitness Application is outlined. The planned goals of the testing were to ensure the system functionality, usability, stability and reliability, with special focus on the Proof of Concept (POC) functionality.

Testing Approach

The testing plan followed a Black Box Testing paradigm, in which the system was tested according to the functional requirements with no consideration on the internal code structure. Automated and manual testing methods were also used to have a complete coverage of the application.

2.1 Black Box Testing

Frontend tests were validated using Black Box Testing which included user registration, profile settings, selection of goal, generation of a meal plan, interaction with the chatbot and navigation through the dashboard. The user interface was used to provide inputs and compare the output with predefined expected result.

Types of Testing Performed

Following shows the different types of testing that were conducted as part of system validation:

3.1 Functional Testing

Functional testing was part of the solution to ensure that every application feature is functional as per requirements. This encompasses:

• User registration and input authentication.

Health information-selection: Tier-2

Goal selection, and goal validation.

• Generation of meal plans and nutritional breakdown.

• Chatbot interaction

The flow of Dashboard and navigation.

3.2 Validation Testing

Validation testing was used to test the response of system in relation to invalid or missing inputs and included:

False or negative values: This option causes any negative or unrealistic value to be entered.

• Filling in of required fields that have no content.

Seeking nothing further through persisting with what was begun.

Submission of an image without uploading a picture of a meal

3.3 User Interface (UI) Testing

UI testing was done to ensure that all objects of the interface such as buttons, forms, labels and screens behave as intended and they also give the user correct feedback.

3.4 Stability Testing

Stability testing was done to determine that the application would have operational integrity under the normal usage conditions. Page refreshing situations were also checked to make sure that the application does not crash and does not lose important data.

3.5 Responsiveness Testing

The responsiveness testing ensured that the application responded appropriately to the different screen sizes, especially the mobile ones to ensure that the user experience remains consistent.

Selenium, Automation testing.

4.1 Tool Selection

The reason why the frontend automation was chosen to be Selenium WebDriver was the broad support of browser and the ability to interact with the latest web technologies. PyTest was the test execution framework that was used to systematize and implement test cases effectively.

4.2 Automation Scope

Automated testing was applied to all frontend functionalities that do not require hardware interaction, including:

- User registration and profile setup
- Goal selection and validation
- Meal plan generation
- Chatbot interaction
- Dashboard and navigation
- UI validation and responsiveness

4.3 Limitations of Automation

Some of the features like camera access, live workout pose detection, and posture real-time feedback could not be automated using Selenium due to the limitations of hardware and limitations of the real-time video stream. The functionalities were thus manually tested.

Manual Testing

The features were tested manually that need real-time user interaction or access to hardware, i.e. the following:

• Camera permission handling

• Live workout pose detection

• Accurate and inaccurate feedback of posture.

These tests guaranteed application behaviour in realistic operating conditions and it is stable even when permissions are denied.

PyTest Selenium Automation Test Case Table

Test Case ID	Module	Test Scenario	PyTest Function Name	Automation Type	Status
TC-01	App Launch	Open application	test_TC01_app_launch()	Automated	Yes
TC-02	Registration	Valid user input	test_TC02_valid_registration()	Automated	Yes
TC-03	Registration	Invalid age input	test_TC03_invalid_age()	Automated	Yes
TC-04	Registration	Empty required fields	test_TC04_empty_fields()	Automated	Yes
TC-05	Health Info	Select health condition	test_TC05_health_condition()	Automated	Yes
TC-06	Health Info	Select allergies	test_TC06_allergy_selection()	Automated	Yes
TC-07	Profile Setup	Submit complete profile	test_TC07_profile_submit()	Automated	Yes
TC-08	Goal Selection	Weight loss goal	test_TC08_weight_loss_goal()	Automated	Yes
TC-09	Goal Selection	Weight gain goal	test_TC09_weight_gain_goal()	Automated	Yes
TC-10	Goal Validation	Unrealistic target weight	test_TC10_unrealistic_weight()	Automated	Yes
TC-11	Goal Validation	No goal selected	test_TC11_no_goal_selected()	Automated	Yes

TC-12	Metrics Display	BMI, BMR, TDEE	test_TC12_metrics_display()	Automated	Yes
TC-13	Metrics Display	Health status label	test_TC13_health_status()	Automated	Yes
TC-14	Meal Plan	Generate meal plan	test_TC14_generate_meal_plan()	Automated	Yes
TC-15	Meal Plan	Nutrition breakdown	test_TC15_nutrition_breakdown()	Automated	Yes
TC-16	Meal Plan	Condition-aware meals	test_TC16_condition_aware_meal()	Automated	Yes
TC-17	Meal Image Upload	Upload food image	test_TC17_food_image_upload()	Partially Automated	Yes
TC-18	Meal Image Upload	Non-food image	test_TC18_non_food_image()	Automated	No
TC-19	Meal Image Upload	Calorie estimation	test_TC19_calorie_estimation()	Automated	Yes
TC-20	Cheat Meal	High-calorie upload	—	N/A	N/A
TC-21	Meal Image Upload	Submit without image	test_TC21_submit_without_image()	Automated	Yes
TC-22	Workout	Open workout screen	test_TC22_workout_screen()	Automated	Yes

TC-23	Workout	Allow camera permission	—	Manual	N/A
TC-24	Workout	Deny camera permission	—	Manual	N/A
TC-25	Pose Detection	Body landmarks	—	Manual	N/A
TC-26	Pose Feedback	Incorrect posture	—	Manual	N/A
TC-27	Pose Feedback	Correct posture	—	Manual	N/A
TC-28	Chatbot	Open chatbot	test_TC28_chatbot_open()	Automated	Yes
TC-29	Chatbot	Send valid query	test_TC29_chatbot_response()	Automated	Yes
TC-30	Chatbot	Empty message	test_TC30_empty_chat()	Automated	Yes
TC-31	Dashboard	View progress	test_TC31_dashboard()	Automated	Yes
TC-32	Navigation	Screen navigation	test_TC32_navigation()	Automated	Yes
TC-33	Stability	Page refresh	test_TC33_page_refresh()	Automated	Yes
TC-34	Responsiveness	Mobile view	test_TC34_responsive()	Automated	Yes

TC-35	POC Completion	Complete POC flow	test_TC35_poc_completion()	Automated	Yes

Manual cases (TC-23 → TC-27) are **tested manually** and not included in automation scripts.

Conclusion

In this chapter, the authors have described a thorough testing methodology that was used to test the AI-based Health and Fitness Application. With the help of automated testing with Selenium and PyTest and manual testing of features that required hardware, exhaustive testing of the system was done. The test cases passed the test and it can be said that the system is reliable, stable and ready to be developed further than in the POC stage.