User Management Web System

Web app

[Task 2]

Mohamad Abdulmoula,R. Rama

ramamohameed9@gmail.com

Department of Software Engineer

Under the Supervision Of:

Eng Asmaa Duramae

July 26, 2025

Table of Contents

Glossary of Terms

Term	**Definition**
Django	A high-level Python web framework that promotes rapid development and clean design.
SQLite3	A lightweight, file-based relational database used for persistent local storage.
Ngrok	A tunneling tool that exposes local servers to public URLs for external access/testing.
Jira	A popular project and issue-tracking software used for agile development.
Xray	A test management tool integrated with Jira to manage test cases, executions, and traceability.
Functional Testing	Testing that ensures each function of the software application operates according to the requirement.
System Testing	End-to-end testing of the integrated software system to validate its compliance with requirements.
End-to-End Testing	A method to test the complete flow of an application from start to finish simulating user behavior.
Black-box Testing	A testing technique that examines the functionality of an application without peering into its internal structures.
MVC Architecture	A software design pattern (Model-View-Controller) used to separate concerns and organize code structure.

Project Overview

This project involves designing a Django-based web application for managing user data. It includes features such as data entry through a form, viewing submitted data in a table, and toggling user status dynamically.

Planning

A. Objective:
To design and implement a Django-based web system that allows users to be added with their name and age, and toggle their status (active/inactive) through a user interface. The application supports data persistence and live status control through web interaction.

B. Purpose:
The system is intended as a lightweight user management interface for educational or prototype systems. It provides a foundational framework for database-driven input, dynamic interface updates, and remote usability testing, serving as a model for future scalable administrative dashboards.

C. Tools Used: Django 5.2.4 ,SQLite3 ,Ngrok ,Python 3.11 ,HTML/CSS ,Jira + Xray

Requirements Analysis

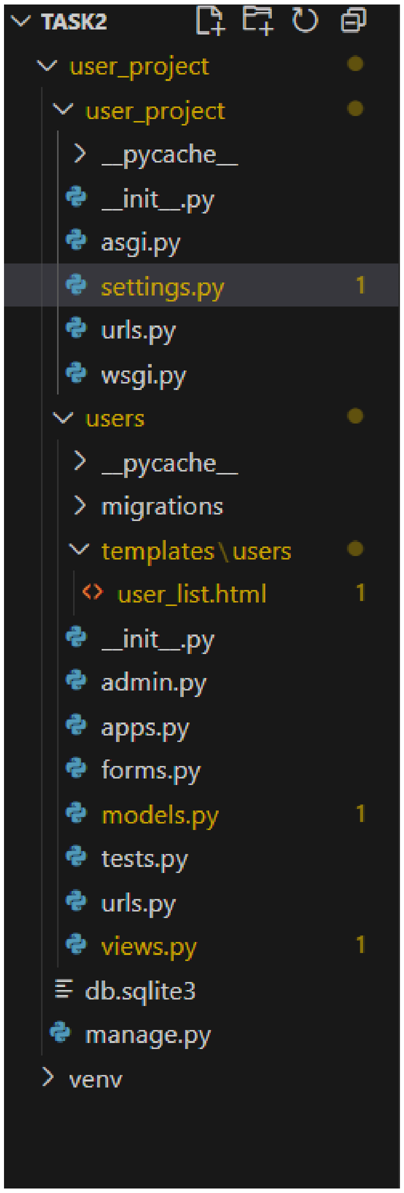
A. **Functional Requirements (Boilerplate Style)**

ID	Requirement Description	Priority	Source
FR1	The system shall allow users to input name and age using a form	High	Instructor
FR2	The system shall store form input into an SQLite3 database	High	Developer
FR3	The system shall display all users in a tabular format on the same page	Medium	Developer
FR4	The system shall allow toggling the user status between 0 and 1 via a button	High	Instructor
FR5	The system shall update and reflect the status toggle in real time	High	Developer

B. **Non-Functional Requirements**

- The system must use Django 5.2.4 or later.
- The UI should be minimalistic and responsive.
- The system should allow access through Ngrok for external testing.

Coding



```
TASK2
  user_project
    user_project
      __pycache__
      __init__.py
      asgi.py
      settings.py        1
      urls.py
      wsgi.py
    users
      __pycache__
      migrations
      templates\users
        user_list.html   1
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py          1
      tests.py
      urls.py
      views.py           1
    db.sqlite3
    manage.py
  venv
```

Figure 1 Django Project Directory Structure for User Management System

This screenshot displays the organized folder structure of the Django project user_project, including the core configuration files (settings.py, urls.py), app-specific files under users, HTML templates for UI rendering, and Python modules responsible for models, views, and forms. It illustrates how the implementation followed Django's MVC architecture for modular and scalable development.

System Testing Phase

A. Testing Objective

To validate the complete functionality, integration, and data persistence of the user management and object recognition system by simulating real world user scenarios through end to end test flows.

B. Testing Approach

1. Strategy

We adopted a **manual black box testing** approach that evaluates the system's external behavior without inspecting its internal logic. The goal is to ensure that user interactions yield the expected outputs under various scenarios.

2. Types of Testing Conducted

A. **Functional-Testing**
Verified that individual features such as form submission, user display, and status toggling operate according to specifications.

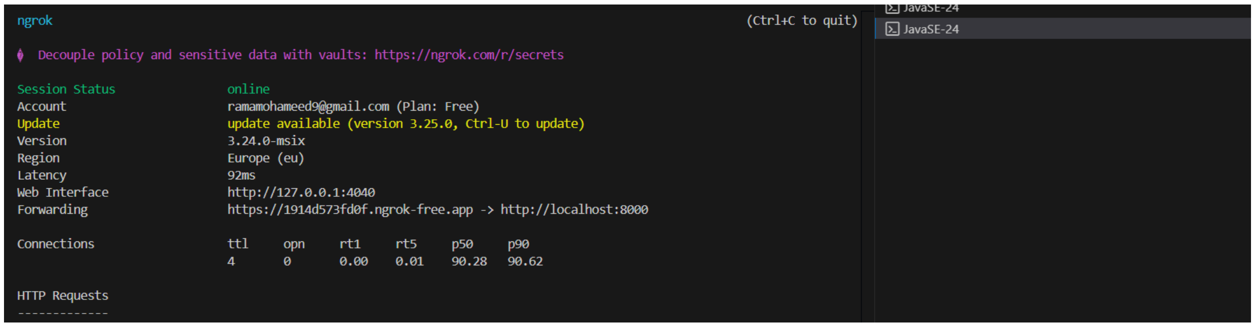
B. **System-Testing**
Validated the full system integration by executing complete user flows from data entry to result persistence and UI update.

C. **End-to-End-Testing (E2E) via Jira/Xray**

Conducted comprehensive user simulations:

a. Adding a user (name + age)
b. Confirming appearance in the user table
c. Toggling the status
d. Refreshing to verify persistence All steps were executed and documented in Xray with attached evidence and result tracking.

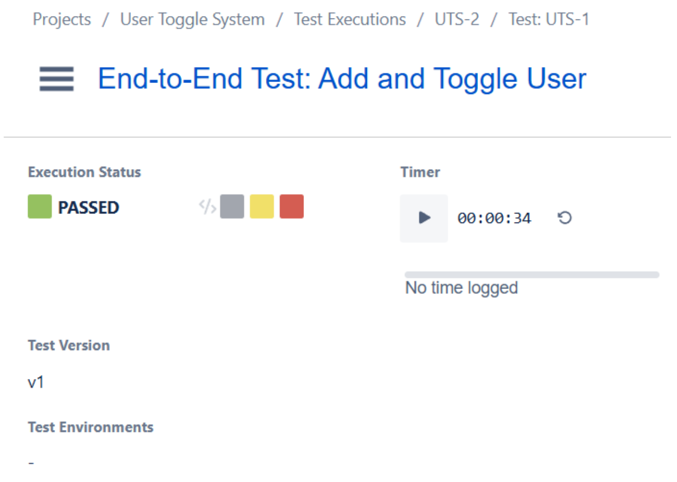
D. Traceability and Evidence



```
ngrok                                                          (Ctrl+C to quit)

♦ Decouple policy and sensitive data with vaults: https://ngrok.com/r/secrets

Session Status                online
Account                       ramamohameed9@gmail.com (Plan: Free)
Update                        update available (version 3.25.0, Ctrl-U to update)
Version                       3.24.0-msix
Region                        Europe (eu)
Latency                       92ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://1914d573fd0f.ngrok-free.app -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              4       0       0.00    0.01    90.28   90.62

HTTP Requests
-------------
```

Figure 2 Ngrok Tunnel Session

The screenshot shows the Ngrok tunnel exposing the local development server (http://localhost:8000) to a public URL (https://1914d573fd0f.ngrok-free.app). This enabled remote end to end testing by simulating real world external access to the system during system and functional testing.



Projects / User Toggle System / Test Executions / UTS-2 / Test: UTS-1

End-to-End Test: Add and Toggle User

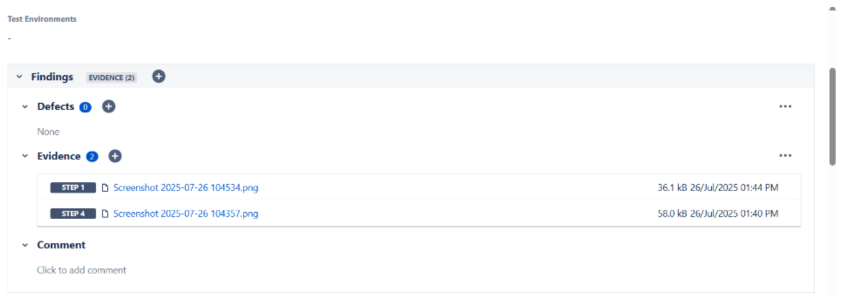
Execution Status
🟩 PASSED

Timer
00:00:34
No time logged

Test Version
v1

Test Environments
-

Figure 3 Test Case Overview in Xray



Test Environments

-

Findings EVIDENCE (2)

Defects 0

None

Evidence 2

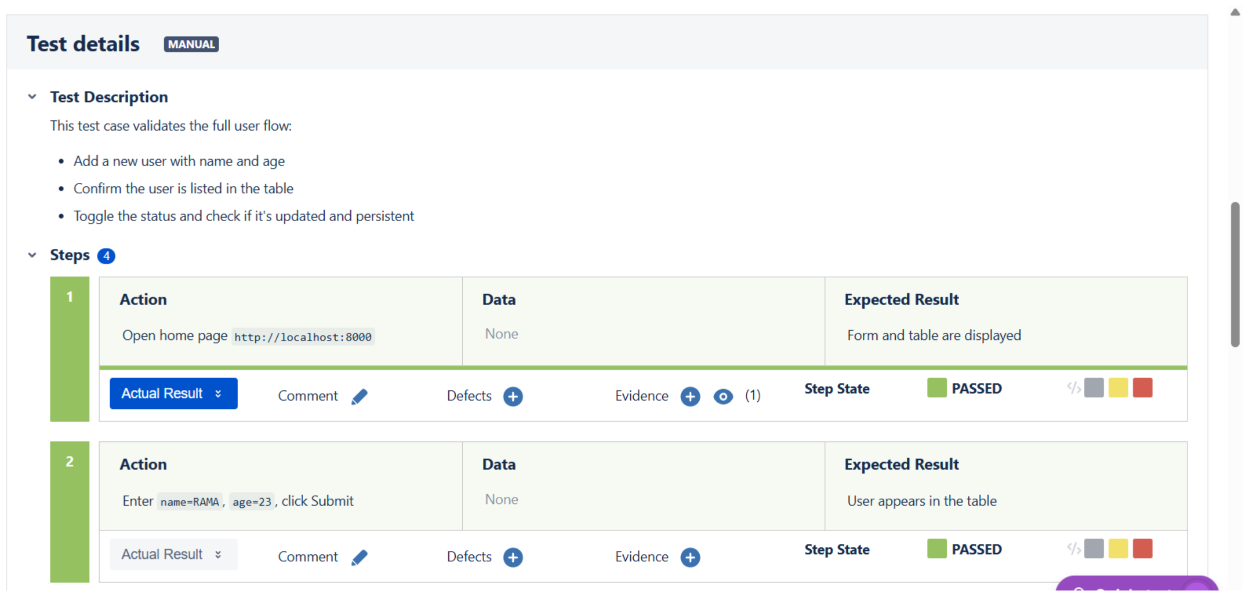
STEP 1	Screenshot 2025-07-26 104534.png	36.1 kB 26/Jul/2025 01:44 PM
STEP 4	Screenshot 2025-07-26 104357.png	58.0 kB 26/Jul/2025 01:40 PM

Comment

Click to add comment

Figure 4 Xray Test Execution Evidence Attachments

This screenshot displays the evidence section of the Xray test execution. It confirms that screenshots for Step 1 (homepage load) and Step 4 (status persistence after refresh) were attached successfully. The presence of timestamps and file sizes enhances traceability and validation of the test outcomes.



Test details MANUAL

Test Description

This test case validates the full user flow:

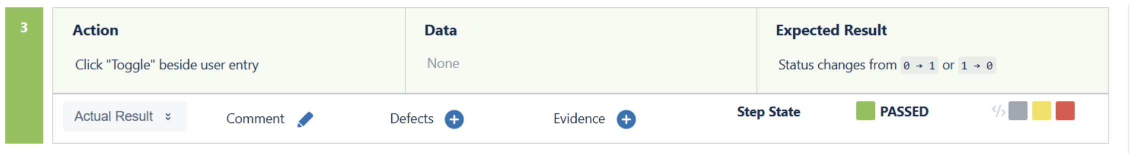
- Add a new user with name and age
- Confirm the user is listed in the table
- Toggle the status and check if it's updated and persistent

Steps 4

	Action	Data	Expected Result
1	Open home page `http://localhost:8000`	None	Form and table are displayed
	Actual Result · Comment · Defects · Evidence (1)		Step State: PASSED
2	Enter `name=RAMA`, `age=23`, click Submit	None	User appears in the table
	Actual Result · Comment · Defects · Evidence		Step State: PASSED

Figure 5 Test Steps Execution in Xray (Steps 1 & 2)

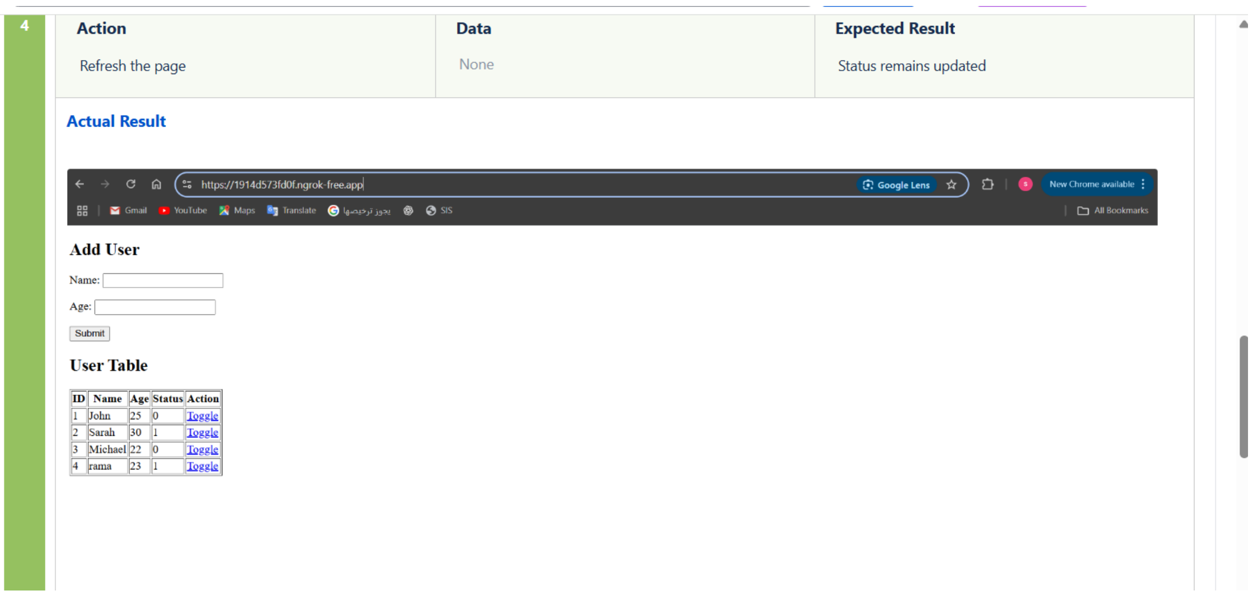
This image shows the initial steps of the end to end test case within Xray. The actions include launching the homepage and submitting a new user (RAMA, 23). Each step shows the expected results, status as PASSED, and attached evidence to confirm functional accuracy.



	Action	Data	Expected Result
3	Click "Toggle" beside user entry	None	Status changes from `0 → 1` or `1 → 0`
	Actual Result · Comment · Defects · Evidence		Step State: PASSED

Figure 6 Toggle Action Verification (Step 3)

This figure highlights the action of toggling the user status. It verifies that the status field updates from 0 → 1 or 1 → 0 and records a PASSED result with tracking for expected behavior during dynamic state change.



4	Action	Data	Expected Result
	Refresh the page	None	Status remains updated

Actual Result

Add User

Name:

Age:

Submit

User Table

ID	Name	Age	Status	Action
1	John	25	0	Toggle
2	Sarah	30	1	Toggle
3	Michael	22	0	Toggle
4	rama	23	1	Toggle

Figure 7 Final Verification and UI Status Persistence (Step 4)

The image shows the refreshed web UI after all test steps were completed. It confirms that the user table remains consistent, the status persists correctly, and the user list includes all submitted entries proving end to end state persistence.

Conclusion

This project successfully demonstrates the design, development, and validation of a lightweight user management system using Django. The web application supports core functionalities such as form-based data entry, dynamic status toggling, and real-time UI updates, all while maintaining data persistence in a local database.

The planning and implementation strictly followed SDLC principles, from requirement analysis using a structured boilerplate format to system testing documented through Xray and Jira. End-to-end testing scenarios were executed to simulate real-world interactions, and remote testing was made possible through Ngrok tunneling.

The integration of tools such as Django, SQLite, Jira, and Xray ensured a robust development and testing workflow. This system can serve as a foundation for more advanced administrative dashboards and supports scalability for future feature extensions such as authentication, role management, or API integration

References

A. Django Documentation – *The official guide for Django 5.2.4 including tutorials, API references, and deployment strategies*. https://docs.djangoproject.com/en/5.2/

B. SQLite Documentation – *Comprehensive details on SQLite syntax, usage, and integration*. https://sqlite.org/docs.html

C. Ngrok Documentation – *Guidance on installing and configuring Ngrok tunnels for exposing local web applications*. https://ngrok.com/docs

D. Xray for Jira – *Official documentation on test management, traceability, and integration with Jira*. https://docs.getxray.app/

E. Jira Software by Atlassian – *Documentation and guides for issue tracking, project planning, and test execution workflows*. https://support.atlassian.com/jira-software-cloud/