**Research on Testing Methodologies**

**1. Manual Testing vs Automated Testing**

***Manual Testing***

* Human tester checks the app step by step without scripts.
* Example: You log in as an organizer, click “Create Event,” fill in the details, and check if the event shows up.
* Best for: Exploratory testing, UI/UX checks, small projects.

***Automated Testing***

* Computer scripts/tools automatically test features.
* Example: Using **Selenium**, you write a script that opens your platform, clicks “Create Event,” fills in fields, and submits it.
* Best for: Repeated tests, regression testing, large projects.

**2. Functional vs Non-Functional Testing**

***Functional Testing = Tests what the system should do.***

* Example: In your project, when a user clicks “Register for Event,” the system should add their name to the attendee list.
* Questions answered: *Does this button/feature work correctly?*

***Non-Functional Testing = Tests how well the system works.***

* Example: 200 users try to register for the same event at the same time → system must handle the load.
* Other checks: performance, usability, security.

**3. Regression Testing**

* Definition: After adding a new feature, check if the old features still work.
* Example: After you add the “Filter Events” function, test again if “Create Event” and “Register for Event” still work as before.

**4. User Acceptance Testing (UAT)**

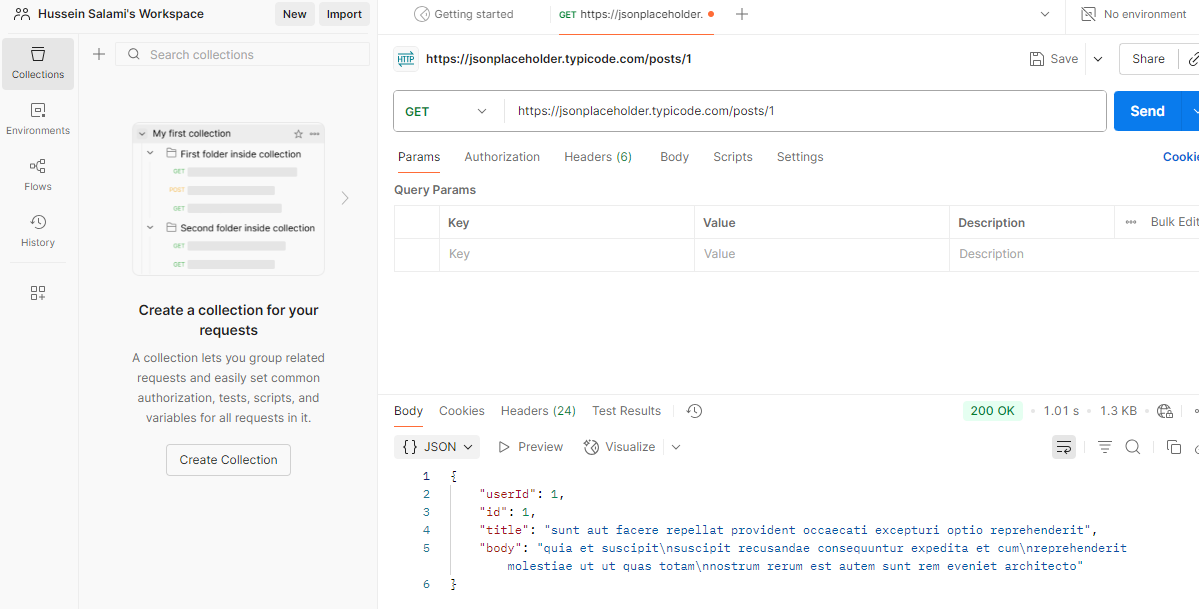
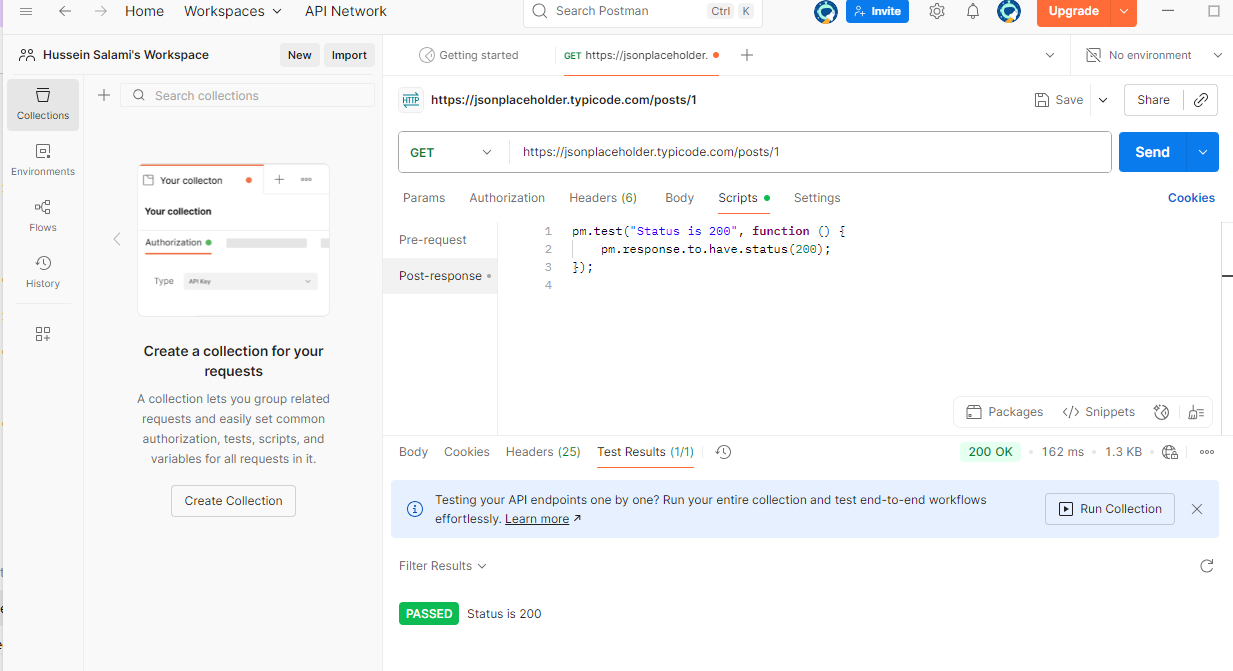
* Definition: Final testing by end-users (real people) before going live.
* Example: Invite 3–4 Darwin residents to test the app. Ask them: “Can you find and register for a cultural event easily?” If they can, UAT passes.

**Research & Write Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | User story | Scenario | Preconditions | steps | Expected result |
| |  | | --- | | **TC-01** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-01 – Event Submission** |  |  | | --- | |  | | | |  | | --- | | Organizer creates an event with valid info . |  |  | | --- | |  | | |  | | --- | | Organizer account verified & logged in |  |  | | --- | |  | | |  | | --- | | 1) Click **Create Event**  2) Fill all required fields  3) Click **Submit** |  |  | | --- | |  | | |  | | --- | | Event status becomes **Submitted**; appears in **Admin Review**; organizer sees success message |  |  | | --- | |  | |
| |  | | --- | | **TC-02** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-02 – Manage Events (Edit)** |  |  | | --- | |  | | | |  | | --- | | Organizer edits a **Draft/Submitted** event and saves |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer has an event in **Draft** (or **Submitted**, not Approved) |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | 1) Open **My Events**  2) Open event  3) Click **Edit**  4) Change Title  5) **Save** |  |  | | --- | |  | | | |  | | --- | | Changes saved; event keeps its state (Draft/Submitted); updated title shown in list/details |  |  | | --- | |  | |
| |  | | --- | | **TC-03** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-02 – Manage Events (Delete)** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer deletes a **Draft** event with confirmation |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer has a **Draft** event |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 1) Open **My Events** 2) Select Draft event  3) Click **Delete**  4) Confirm |  |  | | --- | |  | | |  | | --- | | Event removed from list; confirmation message displayed; no longer searchable |  |  | | --- | |  | |
| |  | | --- | | **TC-04** |  |  | | --- | |  | | |  | | --- | | **US-03 – Organizer Profile** |  |  | | --- | |  | | |  | | --- | | Organizer updates profile and requests verification |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer logged in; profile not verified |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 1) Open **Profile**  2) Upload logo  3) Enter website/contact  4) Click **Request Verification** |  |  | | --- | |  | | |  | | --- | | Profile saved; status = **Pending Verification**; organizer sees “Submitted for review” message |  |  | | --- | |  | |
| |  | | --- | | **TC-05** |  |  | | --- | |  | | |  | | --- | | **US-04 – Browse Events** |  |  | | --- | |  | | |  | | --- | | Attendee filters by **Category=Family** and **Date=Weekend** |  |  | | --- | |  | | |  | | --- | | ≥3 events exist with mixed categories/dates |  |  | | --- | |  | | |  | | --- | | 1) Open **Events**  2) Set filters (Family + Weekend)  3) Click **Apply** |  |  | | --- | |  | | |  | | --- | | Only matching events listed; count updates; no unrelated events shown |  |  | | --- | |  | |
| |  | | --- | | **TC-06** |  |  | | --- | |  | | |  | | --- | | **US-05 – Event Details** |  |  | | --- | |  | | |  | | --- | | Attendee views event details page (map/time/contact visible) |  |  | | --- | |  | | |  | | --- | | Event has location, time, contact info |  |  | | --- | |  | | |  | | --- | | 1) From list, open the event card  2) View details |  |  | | --- | |  | | |  | | --- | | Page shows title, description, images, **map**, **local time**, and **contact info** correctly |  |  | | --- | |  | |
| |  | | --- | | **TC-07** |  |  | | --- | |  | | |  | | --- | | **US-06 – Save & Share** |  |  | | --- | |  | | |  | | --- | | Attendee adds event to **Google Calendar** successfully |  |  | | --- | |  | | |  | | --- | | Event is public; user in browser session |  |  | | --- | |  | | |  | | --- | | 1) Open event details  2) Click **Add to Calendar**  3) Choose **Google** 4) Confirm |  |  | | --- | |  | | |  | | --- | | Calendar entry created with correct title, start/end time, and venue |  |  | | --- | |  | |
| |  | | --- | | **TC-08** |  |  | | --- | |  | | |  | | --- | | **US-07 – Review Submissions** |  |  | | --- | |  | | |  | | --- | | Admin **approves** a submitted event with a note |  |  | | --- | |  | | |  | | --- | | At least one event in **Submitted** state |  |  | | --- | |  | | |  | | --- | | 1) Login as **Admin** 2) Open **Review Queue**  3) Open event  4) Click **Approve**  5) Add note  6) Confirm |  |  | | --- | |  | | |  | | --- | | Status = **Approved**; event becomes public; organizer notified with the note |  |  | | --- | |  | |
| |  | | --- | | **TC-09** |  |  | | --- | |  | | |  | | --- | | **US-08 – Content Moderation** |  |  | | --- | |  | | |  | | --- | | Admin removes an **inappropriate** public event |  |  | | --- | |  | | |  | | --- | | Event is public and flagged as inappropriate |  |  | | --- | |  | | |  | | --- | | 1) Admin opens event  2) Click **Remove/Unpublish** 3) Enter reason  4) Confirm |  |  | | --- | |  | | |  | | --- | | Event hidden/unpublished; reason logged; no longer visible to attendees |  |  | | --- | |  | |
| |  | | --- | | **TC-10** |  |  | | --- | |  | | |  | | --- | | **US-09 – Organizer Verification** |  |  | | --- | |  | | |  | | --- | | Admin verifies organizer after document check (ABN/community proof) |  |  | | --- | |  | | |  | | --- | | Organizer submitted verification; docs provided |  |  | | --- | |  | | |  | | --- | | 1) Admin opens **Organizer Verification** queue 2) Review docs  3) Click **Verify** |  |  | | --- | |  | | Organizer marked **Verified**; organizer notified; verified badge appears on profile/events |

**Learn Automation Tool**

Installed Postman:



Completed Course about Postman: [file:///C:/Users/hp/Downloads/CertificateOfCompletion\_Introducing%20Postman.pdf](file:///C:\Users\hp\Downloads\CertificateOfCompletion_Introducing%20Postman.pdf)

**Identify Testing Tools**

**Postman**

* What it is: A tool to check if APIs (the “back-end connections”) are working.
* What it does: You can send requests like “give me all events” and see the reply from the system.
* Simple example: Type the event API link in Postman → click Send → it shows event details in JSON format.

**Selenium**

* What it is: A tool to control a web browser automatically.
* What it does: You write small scripts that click buttons, type in forms, and check results.
* Simple example: Selenium opens Chrome, logs in as an organizer, fills out the event form, clicks submit, and checks that the event was created.

**JMeter**

* What it is: A tool to test performance (speed under heavy load).
* What it does: Pretends to be many users using the system at the same time.
* Simple example: You set 200 fake users in JMeter → all apply filters at once → JMeter measures how fast the system responds.

**Fiddler**

* What it is: A tool to watch and record network traffic (the messages going between your computer and the server).
* What it does: Helps you see what’s sent and what’s received.
* Simple example: You click “Register for Event” → Fiddler shows the exact request your browser sent and the server’s reply.

**Cypress**

* What it is: A modern tool for testing websites from start to finish.
* What it does: Runs in the browser and checks that pages, buttons, and flows work correctly.
* Simple example: Cypress opens your site, applies “Family Weekend” filter, and checks that only Family events show in the list.