



Hands UP

Testing Policy

updated: 28/09/2025

Contents

1	Introduction	3
2	Unit Testing (Backend and Frontend)	3
2.1	Objective	3
2.2	Automation	3
2.3	Framework	3
3	Integration Testing	4
3.1	Objective	4
3.2	Automation	4
3.3	Framework	4
4	Quality Assurance	5
4.1	Performance Testing	5
4.1.1	Objective	5
4.1.2	Simulation Environment	6
4.1.3	Metrics	6
4.1.4	Metric Weighting	6
4.1.5	Result Scoring	6
4.2	Availability	8
4.2.1	Objective	8
4.2.2	Simulation Environment	9
4.2.3	Metrics	9
4.3	Usability Testing	10
4.3.1	Objective	10
4.3.2	Methodology	10

1 Introduction

This document outlines the **Testing Policy** of HandsUp. The purpose of this policy is to define the structure of our testing procedures to ensure that the application is thoroughly validated and works as intended.

Our testing strategy includes the following:

- Unit Testing (Backend and Frontend) with Jest
- Integration Testing with Cypress
- Performance Testing with Lighthouse
- User Testing for usability and feedback

The following sections describe the tools, procedures, and standards we applied to achieve reliable, high-quality results.

2 Unit Testing (Backend and Frontend)

2.1 Objective

The objective of unit testing is to validate individual components in isolation, ensuring that each small, independent part of the system functions correctly.

2.2 Automation

All unit tests are automated and seamlessly integrated into the **GitHub Actions pipeline**, ensuring that every change is validated before being merged into the development branch.

2.3 Framework

We use **Jest** to implement automated unit testing across both backend and frontend components.

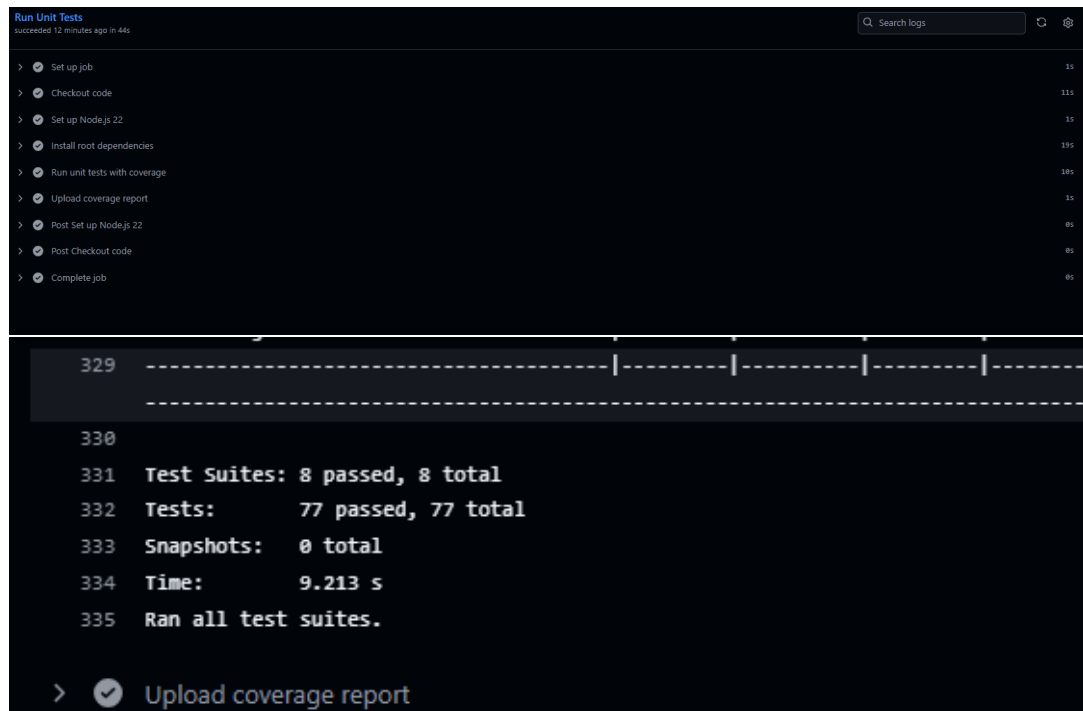


Figure 1: Example of GitHub Actions tests passing

3 Integration Testing

3.1 Objective

Integration tests validate the interactions between different system components, ensuring they work together as expected.

3.2 Automation

All integration tests are automated and integrated into the **GitHub Actions pipeline**, guaranteeing that all parts of the system are functioning correctly before merging code.

3.3 Framework

We use **Cypress** to conduct integration tests, validating the user interface and user interactions.

```

Cypress Integration Tests
succeeded 6 minutes ago in 4m 39s

> Set up job
> Checkout
> Set up Node.js 22
> Install docker-compose
> Create .env file (for local testing/debugging, though direct env injection is better for CI)
> Start database services
> Wait for PostgreSQL database to be ready
> Run database migrations (and skip on failure)
> Install root dependencies
> Install backend API dependencies
> Install frontend dependencies
> Start services (backend API and frontend)
> Wait for application services to be ready
  Show service logs if wait failed
> Run Cypress tests
  Upload Cypress artifacts
> Post Set up Node.js 22
> Post Checkout
> Complete job

260
261
262 tput: No value for $TERM and no -T specified
263
264 Spec                                Tests  Passing  Failing  Pending  Skipped
265 | ✓ home.cy.js                        00:05   1       1       -       -       - |
266 |
267 | ✓ learn.cy.js                      01:08   9       9       -       -       - |
268 |
269 | ✓ signUpLogin.cy.js                00:10   7       7       -       -       - |
270 |
271 | ✓ translate.cy.js                 00:49  13      13       -       -       - |
272 |
273 | ✓ userProfile.cy.js               00:08   1       1       -       -       - |
274 |
275 | ✓ All specs passed!               02:22  31      31       -       -       - |
276
  Upload Cypress artifacts

```

Figure 2: Example of GitHub Actions tests passing

4 Quality Assurance

For HandsUp, **quality assurance** is critical, as user experience and performance directly influence how the application is perceived. To maintain a high standard, we placed strong emphasis on **performance testing** and **usability testing**.

4.1 Performance Testing

4.1.1 Objective

Performance testing evaluates the speed, responsiveness, and stability of the application under various conditions, ensuring that it can handle expected loads efficiently.

4.1.2 Simulation Environment

To simulate real-world usage, we throttled CPU and network conditions to approximate a mid-tier device on a 3G/4G connection. This ensures results reflect typical user experiences rather than high-end developer machines.

4.1.3 Metrics

We used Lighthouse to measure the following key performance indicators:

- First Contentful Paint (FCP)
- Largest Contentful Paint (LCP)
- Speed Index (SI)
- Time to Interactive (TTI)
- Total Blocking Time (TBT)
- Cumulative Layout Shift (CLS)

4.1.4 Metric Weighting

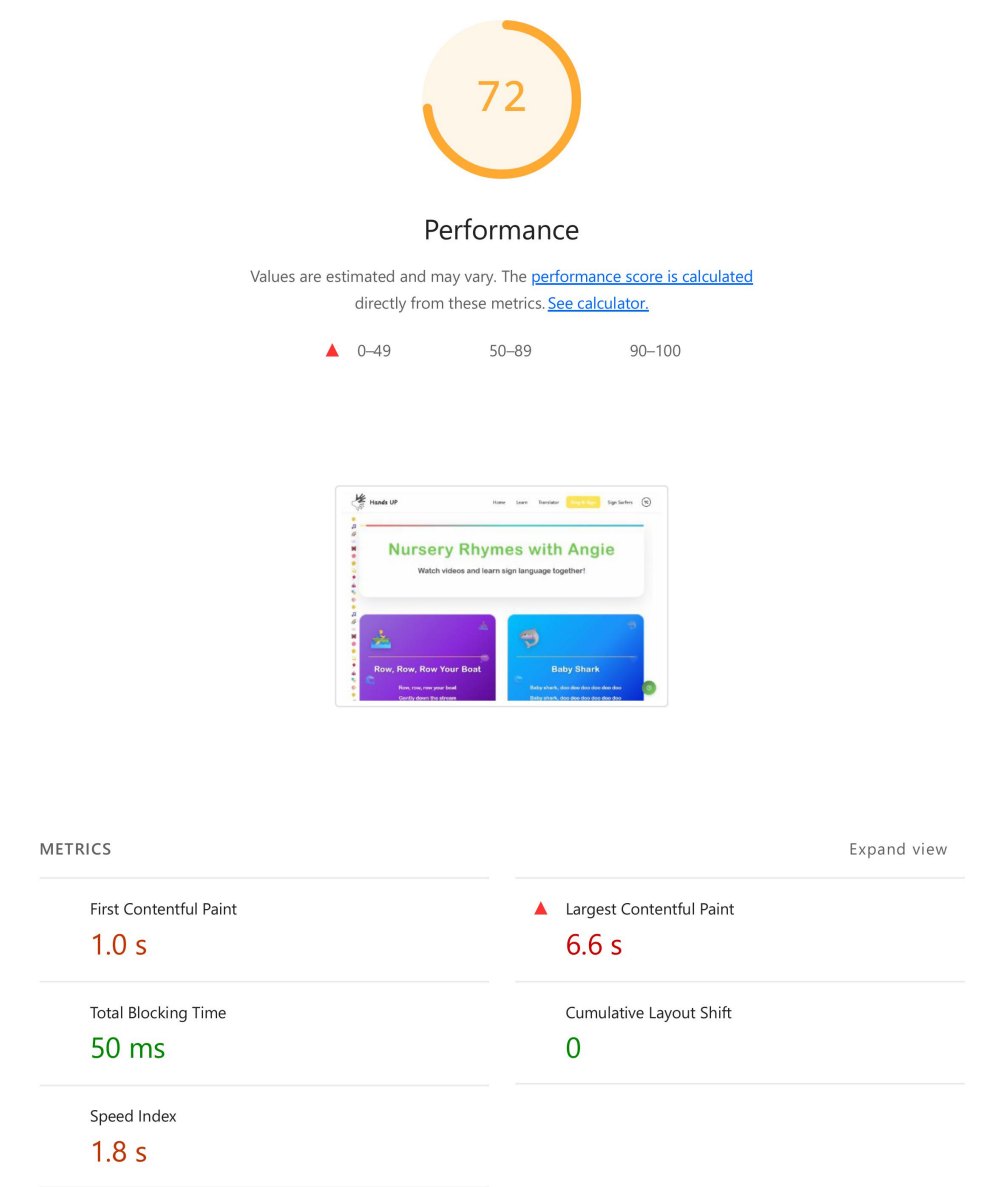
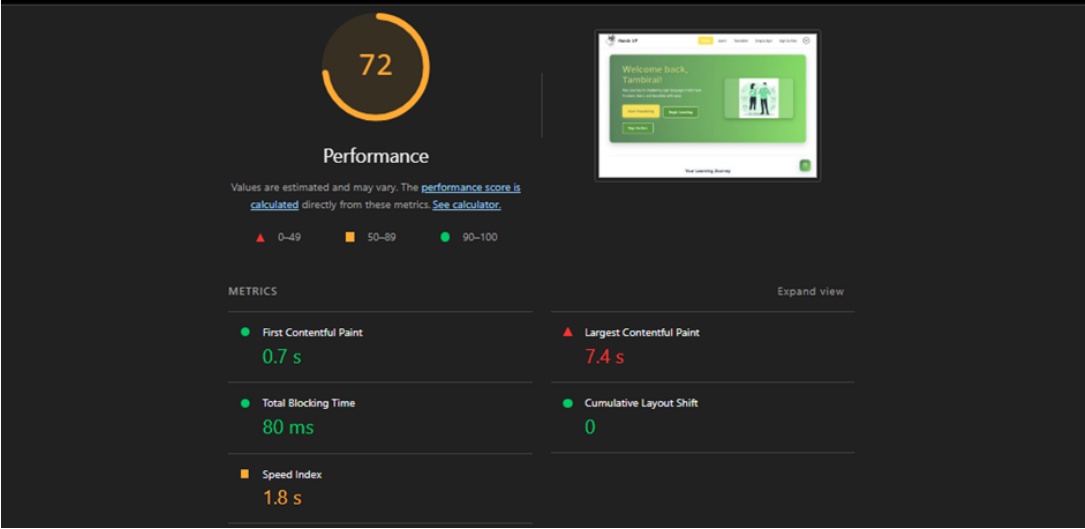
Each metric contributes differently to the overall Lighthouse performance score:

- LCP: ~25%
- TBT: ~30%
- CLS: ~15%
- FCP: ~10%
- SI: ~10%
- TTI: ~10%

4.1.5 Result Scoring

Results are normalized on a scale from 0–100 using a log-normal curve. This prevents small delays from being overly penalized while still highlighting poor performance.

For HandsUp, we aimed for a performance score between **70–100**, balancing graphical features, “wow factors,” and the translator page.



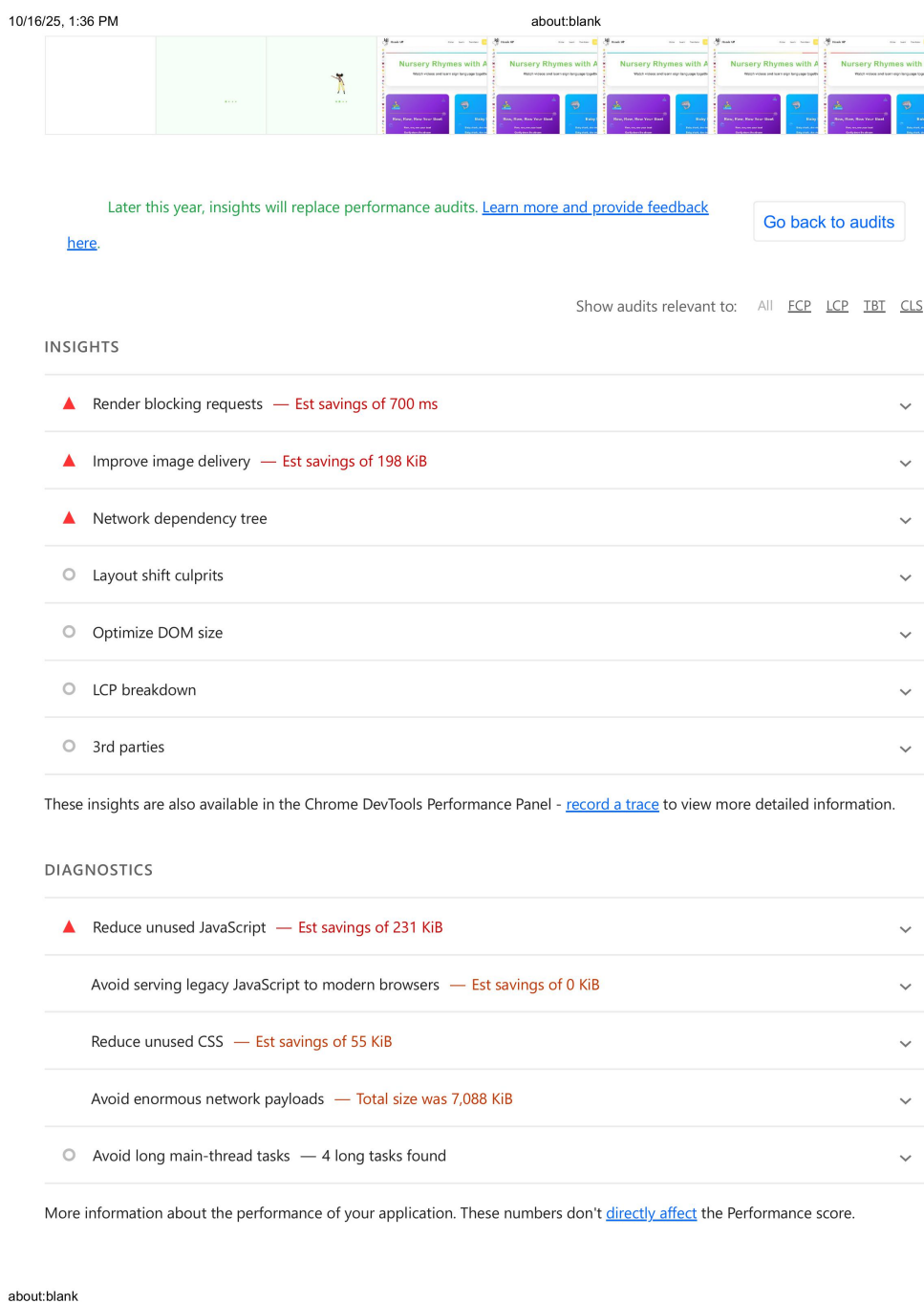


Figure 3: Lighthouse performance report example

4.2 Availability

4.2.1 Objective

Ensure that HandsUp remains up and responsive under normal and peak user load, without unexpected downtime.

4.2.2 Simulation Environment

We use **Uptime Kuma** to continuously monitor the availability of our site. The HandsUp application endpoint is added as a monitored target within Uptime Kuma. From there:

- Uptime Kuma sends requests to the application at regular intervals (configurable, typically every 60 seconds).
- If the application fails to respond within the timeout threshold, the system records a downtime event.
- Monitoring is performed externally, simulating a real user trying to reach the app.

4.2.3 Metrics

Our target uptime is **99%** during non-deployment periods. Uptime Kuma helps us achieve this by:

- Tracking response successes/failures and compiling uptime percentages.
- Recording the response time (latency) of each check, allowing us to see performance degradation before full outages occur.
- Maintaining historical logs of incidents and recovery times.
- Providing real-time notifications (e.g., via email, Discord, or Slack) when downtime is detected so issues can be addressed quickly.

This provides not only raw uptime statistics but also actionable insight into **when**, **why**, and **how long** the system may have been unavailable.

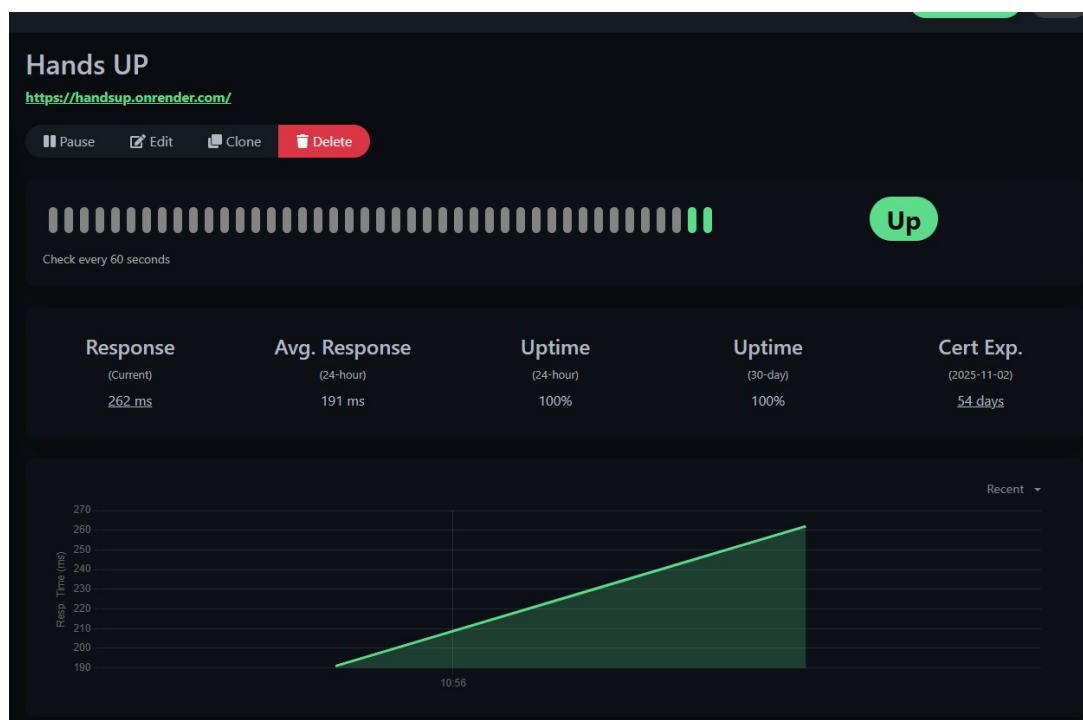


Figure 4: Example of Uptime Kuma monitoring dashboard

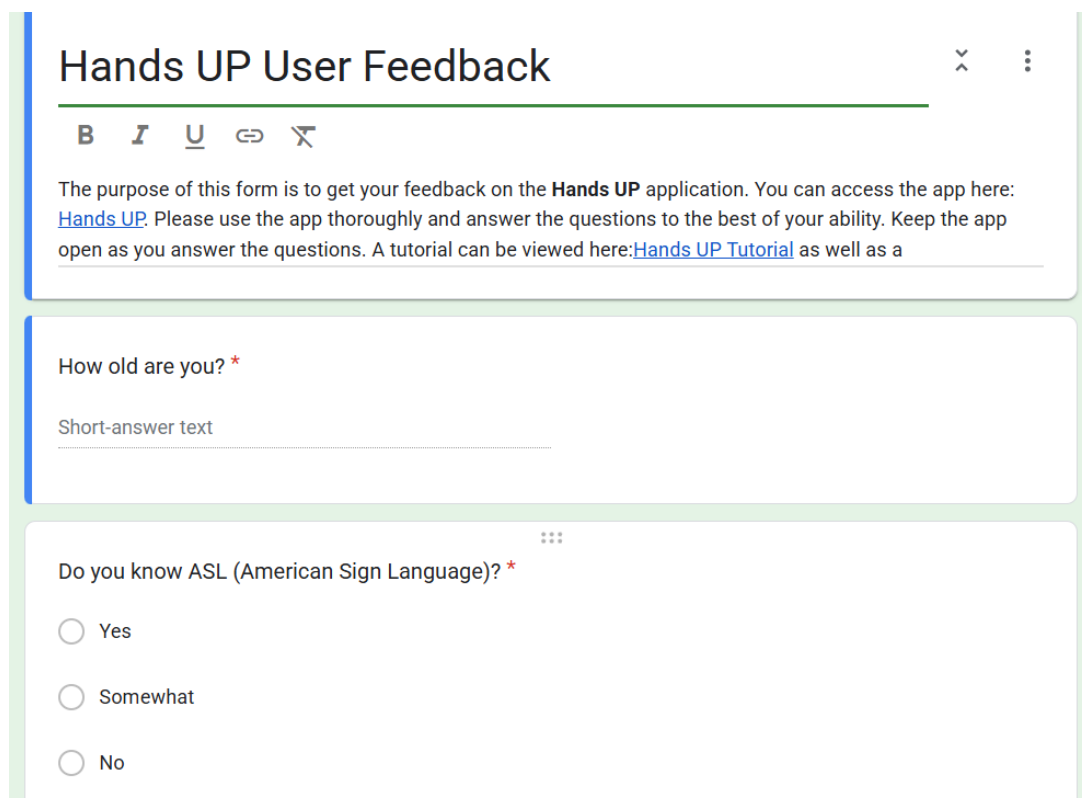
4.3 Usability Testing

4.3.1 Objective

Usability testing ensures that HandsUp is **intuitive, clear, and accessible** to all users. The main goal is to collect feedback from people outside the project team to guide improvements.

4.3.2 Methodology

- **Pre-Planing:** We created a Google Form for participants to provide real-time feedback while interacting with the app.
- **Implementation:** Team members engaged with different user groups to reduce bias. Participants filled out the form, and responses were reviewed collectively. Here is a link to the Form we used: **HandsUp Usability Feedback Form**
- **Follow-Up:** Based on the feedback, we refined or discussed improvements for specific features of the app.



The screenshot shows a Google Form titled "Hands UP User Feedback". The form has a title bar with a close button (X) and a menu button (three dots). Below the title bar is a text area with formatting options (B, I, U, link, unlink). The text in the area reads: "The purpose of this form is to get your feedback on the **Hands UP** application. You can access the app here: [Hands UP](#). Please use the app thoroughly and answer the questions to the best of your ability. Keep the app open as you answer the questions. A tutorial can be viewed here: [Hands UP Tutorial](#) as well as a". Below this text area is a question: "How old are you? *". The question has a "Short-answer text" input field. Below the input field is another question: "Do you know ASL (American Sign Language)? *". This question has three radio button options: "Yes", "Somewhat", and "No".

Figure 5: Example of usability testing feedback collection

Section 3 of 7

Learn Page ✕ ⋮

Description (optional)

Did you complete the placement test? *

☐ Yes

☐ No, I skipped it

⋮

If yes, did you find the placement test to be fair?

☐ Yes

☐ No

How easy was it to access the lessons in each category? *

Figure 6: Example of usability testing feedback collection

Section 4 of 7

Translator Page ✕ ⋮

For best results, experiment with the translator more than once and with different models (alphabets, numbers and glosses).

Were you able to successfully translate something you know? *

☐ Yes

☐ No

⋮

If no, please describe any issues or difficulties you experienced.

Short-answer text

Figure 7: Example of usability testing feedback collection