



Master Thesis

**SEE: A Benchmark for VLM-Generated Screen
Descriptions for Visually Impaired Users**

by

Cédric Makhoul
(cma267)

First Supervisor: Vincent François-Lavet
Daily Supervisor: Mark Scholten
Second Reader: Shujian Yu

June 29, 2025

Abstract. Visually impaired users face significant challenges when navigating desktop interfaces. To address this issue, this thesis introduces the SEE (Screen Explanation Evaluation) Benchmark, the first open-source framework to systematically evaluate how well vision-language models (VLMs) can generate clear and comprehensive screen descriptions for visually impaired users. The benchmark consists of 100 manually annotated Dutch desktop screenshots from real-world government, workplace, and daily-use interfaces, each paired with a reference description and a set of essential elements. Evaluation is performed using a custom LLM-as-a-Judge scoring method, enabling scalable and reproducible comparisons of different models and prompting strategies.

Applying the benchmark yielded clear insights: Mixture-of-Experts (MoE) models, notably LLaMA 4 Scout and LLaMA 4 Maverick, consistently outperformed dense models ($\leq 90\text{B}$ parameters), with Maverick producing the highest quality screen descriptions. Additionally, prompting strategies had a significant impact on description quality. In particular, few-shot prompting combined with iterative refinement guided by benchmark feedback produced the best results. To validate the practical value of the benchmark-optimized screen descriptions, a user study was conducted with six blind Dutch government employees. All participants unanimously reported that these VLM-generated descriptions would significantly enhance their sense of independence and screen overview, and strongly indicated that they would like to use such a tool daily for work. These findings demonstrate that the SEE Benchmark provides a reliable foundation for developing effective VLM-based screen description tools for visually impaired users.

Keywords: Screen accessibility · Vision-language models · LLM-as-a-Judge · Benchmarking · Visually impaired users · Assistive technology

Table of Contents

1	Introduction	4
1.1	Background and Motivation	4
1.2	Problem Statement	4
1.3	Objective and Contribution	5
1.4	Research Questions	5
2	Related Work	6
2.1	Screen Readers	6
2.2	Vision-Based AI Tools for the Visually Impaired	6
2.3	Cloud-Based Desktop Solutions	7
3	Methodology	7
3.1	Benchmark Dataset and Annotation Design	7
3.2	Model Selection	10
3.3	Evaluation Methods	10
3.4	Prompt Annotation Feedback Loop	11
3.5	Prompt-Engineering Strategies	12
3.6	User Study	13
4	Experiments and Results	14
4.1	Evaluation of Small-Scale Models	14
4.2	Evaluation of Larger Models using SEE	15
4.3	Feedback-based Prompt	18
4.4	User Study with Blind Users	19
5	Conclusion	21
A	Appendix A: Evaluation Test Cases	23
B	Appendix B: Prompt Variants Used in Model Comparison	24
C	Appendix C: User Study Questions	27
D	Appendix D: SEE Benchmark Source Code and Annotation Dataset	28

1 Introduction

1.1 Background and Motivation

For visually impaired desktop users, navigating digital interfaces is a fundamentally unequal experience. While sighted users scan screens in a top-down fashion to quickly grasp the layout and purpose of an interface, visually impaired users often rely on screen readers [25]. These screen readers present interfaces as a linear sequence of elements [12]. This bottom-up approach requires users to build a mental model of the screen through sequentially announced headings, buttons, and form fields, relying entirely on developer-defined metadata, such as classes, labels, and descriptions. Screen readers are widely used across work, personal, and educational contexts, where they support essential tasks such as reading documents, completing forms, and navigating websites and software applications [6].

However, many aspects of the interface, such as visual grouping, prominence, and overall screen purpose, are rarely explicitly encoded and therefore cannot be directly conveyed by a screen reader. Visual components like charts or images also remain inaccessible unless explicitly described. As a result, the process of gaining a full understanding of what is on the screen requires effort, memory, and inference.

1.2 Problem Statement

There are several interrelated challenges that prevent visually impaired users from efficiently and accurately understanding the contents of a desktop screen.

First, while the bottom-up paradigm used by screen readers has traditionally been the default, it is inherently more effortful than the top-down visual strategy used by sighted users. Understanding emerges slowly through sequential inspection, which often leads to cognitive overload [12].

Second, screen readers rely entirely on developer-supplied metadata. When this metadata is missing, incomplete, or incorrectly applied, a frequent occurrence in complex or custom-built systems, the user receives partial, ambiguous, or misleading output [30]. This can lead to incorrect mental models of what is actually on the screen.

Third, although better labelling practices could address some of these issues, relying on universal compliance is unrealistic. Despite years of accessibility standards, many digital environments remain poorly labelled, especially in work or governmental software [27,16].

Fourth, screen readers themselves are not intelligent systems. They cannot interpret the visual layout, detect visual emphasis, or understand spatial relationships between elements. They process each component in isolation, without any awareness of the interface as a whole. This leaves a significant gap between how interfaces are designed visually and how they are accessed structurally.

To close this gap, a new approach is needed, one that mimics the strategy of sighted users. Instead of relying solely on metadata, a visually intelligent

model can interpret the entire screen, generate a top-down description and relay this to the visually impaired user. This would enable visually impaired users to quickly grasp the screen’s content, reducing the need for linear exploration and significantly lowering cognitive effort compared to traditional screen readers.

1.3 Objective and Contribution

Recent advances in vision-based AI for blind and visually impaired users, particularly in object recognition, scene description, and real-world navigation, have shown that visual interpretation systems can significantly enhance understanding and independence [24,7]. These developments raise the question of whether similar technologies can be applied to digital desktop environments. If a model can help users understand a street scene, it may also be capable of helping them understand the contents of a desktop screen.

To address this need, this thesis investigates whether large vision-language models (VLMs), can serve as visually intelligent systems capable of describing desktop screens. These models have recently shown promising results in tasks like chart summarization, document layout analysis, and image-grounded captioning [28,11,5], indicating a potential to interpret screen content visually, even in the absence of explicit metadata. This makes them a strong candidate for generating top-down screen descriptions.

Additionally, using local, open-source models ensures that the screenshots can be analysed locally, therefore safeguarding possible sensitive on-screen information. This is critical to ensure deployability in professional or governmental settings, as well as protect sensitive information in private use.

While the initial research goal was to design an open-source, locally executable system for producing coherent VLM-generated screen descriptions, development revealed that no standardized method existed to assess performance for this task. Without a benchmark, it was impossible to compare outputs or track meaningful improvements.

As a result, the primary contribution of this thesis became the creation of the SEE (Screen Explanation Evaluation) Benchmark, an open-source evaluation framework based on 100 annotated Dutch desktop screenshots. This benchmark enables, for the first time, systematic comparison of VLM-generated screen descriptions. Using an LLM-as-a-judge approach, where an LLM evaluates outputs based on predefined evaluation criteria, a model–prompt combination was established and validated during a user study, and serves as a foundation for developing VLM-based screen description systems.

1.4 Research Questions

This thesis is guided by the following research questions:

- **RQ1:** Can VLMs generate coherent and accurate screen descriptions that significantly improve desktop interface understanding for visually impaired users?

- **RQ2:** How can the quality of VLM-generated screen descriptions be evaluated in a scalable and reproducible manner?
- **RQ3:** What is the effect of prompt design on the quality of VLM-generated screen descriptions?
- **RQ4:** What model size is required for VLMs to generate accurate and complete screen descriptions?

2 Related Work

2.1 Screen Readers

Screen readers are the most used assistive technology for accessing digital interfaces by visually impaired users. According to WebAIM’s 2024 survey of 1 539 respondents, 89.7% use a screen reader on a desktop or laptop device [26]. The most commonly used primary screen readers are JAWS (40.5%), NVDA (37.7%), and VoiceOver (9.7%). These screen readers function similarly to translate digital content into auditory or Braille outputs. They detect and convey the digital interface through the underlying structural elements, such as headings, buttons, links, and form fields.

However, high usage does not imply reliable accessibility. According to the 2025 WebAIM Million report, 94.8% of the world’s top one million homepages contain measurable WCAG violations, with an average of 51 accessibility errors per page [27,23]. The most frequent failures are: low-contrast text (79.1%), missing `alt` attributes (55.5%), unlabelled form fields (48.2%), and empty links or buttons. Due to the reliance of the screen reader on this metadata, these failures significantly affect its performance, leading to incomplete or misleading output.

These metadata failures result in practical burdens for users, as missing labels significantly increase cognitive load and task completion time [12]. Even expert screen reader users struggle to interpret screens in the absence of accessibility metadata, demonstrating that such issues cannot be resolved by the user themselves [30]. These problems persist despite the European Accessibility Act, which mandates WCAG compliance for all digital services within the European Union [16,23].

2.2 Vision-Based AI Tools for the Visually Impaired

Most research into AI-based assistance for blind users has focused on applying computer vision to real-world scenes [20]. This includes tasks like recognizing objects, reading signs, identifying people, or describing surroundings using a smartphone camera or smart glasses. The goal is typically to enhance situational awareness or enable safe navigation in physical environments [19,21,9,18]. Consequently, much of the academic work and innovation in assistive vision AI has prioritized these physical-world use cases.

In many of these scenarios, the technology is not only present but highly effective. Popular products such as Seeing AI, OrCam, and Envision AI can

recognize objects, read text aloud, describe traffic lights and street signs, and identify faces in real time [19,21,9]. These applications showcase how vision AI has already matured into a practical, everyday support tool for visually impaired users navigating the physical world.

When vision-based AI is applied in digital contexts, it is still predominantly targeted at mobile environments. For example, AI features for Apple VoiceOver or Gemini in Pixel are able to effectively detect and describe screen content [2,13]. can perform image analysis. These tools, however, remain restricted in the mobile scope.

The desktop experience, in contrast, lags far behind. Despite being the primary environment for work-related tasks, desktop interfaces rarely benefit from vision-based assistance. A few mainstream systems apply computer vision to interpret desktop screen contents for blind users, however all through cloud-based solutions, making it inapplicable in work environments. This results in a significant accessibility gap, as this is where complex interfaces, charts, forms, and custom applications are often present and of significant importance.

2.3 Cloud-Based Desktop Solutions

A small number of assistive tools have begun to address the challenge of screen understanding in desktop environments using cloud-based processing. These systems typically operate by capturing a screenshot of the current interface and sending it to a remote server, where an AI model generates a description of the screen. The resulting description is returned as text and read aloud by a screen reader or presented in a separate interface.

For example, JAWS offers a feature called Picture Smart AI that provides screen descriptions through a cloud-based service [10]. Be My Eyes for Windows, enables users to invoke a keyboard shortcut, which captures a screenshot and sends it for remote processing [3]. After a brief delay, a description of the screen is returned to the user.

Although these tools represent an important step toward more intelligent access to screen content, their design remains non-transparent. It is unclear what types of models are used, or how consistent or complete the generated descriptions are. There are no claims on the accuracy of the descriptions, nor are these products comparable in quality due to a lack of a standardized benchmark. Nevertheless, they demonstrate the growing interest in screen understanding tools for visually impaired users and provide early evidence that useful descriptions can be generated automatically.

3 Methodology

3.1 Benchmark Dataset and Annotation Design

In the creation of the SEE Benchmark, a dataset size of 100 real-world desktop screenshots representing work, government and daily scenarios, such as websites,

dashboards, forms, and installation windows were chosen (Appendix D). The goal was to ensure that the dataset encompasses a wide variety of real-world scenarios to correctly measure the quality of VLM-generated screen descriptions, as well as reduce the chance of prompts overfitting on specific benchmark screens. Each screenshot is annotated with a structured JSON format, including the fields shown in Table 1.

Table 1: Annotation fields used in the SEE Benchmark

Field	Description
image_id	Unique identifier for the screenshot
reference_description	Human-written screen description
must_include	List of critical elements that must be included

These reference descriptions were crafted manually, with a focus on delivering clear top-down description of the screen. This meant starting with the screen’s purpose, and following up with actionable elements (like buttons and forms). After noticing the inability of VLMs to differentiate between important and complementary elements on the screen, a visual importance hierarchy was created. For the sake of this research, this was defined as follows:

- The description begins with the type of screen (login, pop-up, etc.).
- If available, the name of the application or website is included.
- Next, the primary purpose or function of the screen is stated.
- The description then lists all elements that are relevant and functionally important to the primary purpose, such as:
 - titles,
 - headings,
 - buttons,
 - input fields,
 - tabs,
 - dropdowns,
 - and other interactive components.
- These elements must be quoted exactly as they appear on the screen.
- If there are multiple items of the same type, at least three examples are named, along with an indication whether more are present (to promote conciseness).
- The structure aims to keep the reference descriptions intentional and complete, making sure that all functionally significant content is captured, without mentioning excessive details.

Additionally, must-include elements were labelled. These entail screen information that is essential to understanding the purpose or content of the screen, such as identifying the organization on a government webpage, or a login button on a login screen. These serve as a critical requirement to the benchmark; if the

model fails to mention these, it will instantly fail that screenshot and receive no points. This was done to add weight to certain aspects of the screen that are more essential, eliminating scenarios in which models receive high scores merely based on semantic similarity, while still missing the essence of the screen.

To illustrate the application of these annotation labels, take an example screenshot from the SEE Benchmark dataset, shown in Figure 1 and Table 2 (both found in Appendix D).

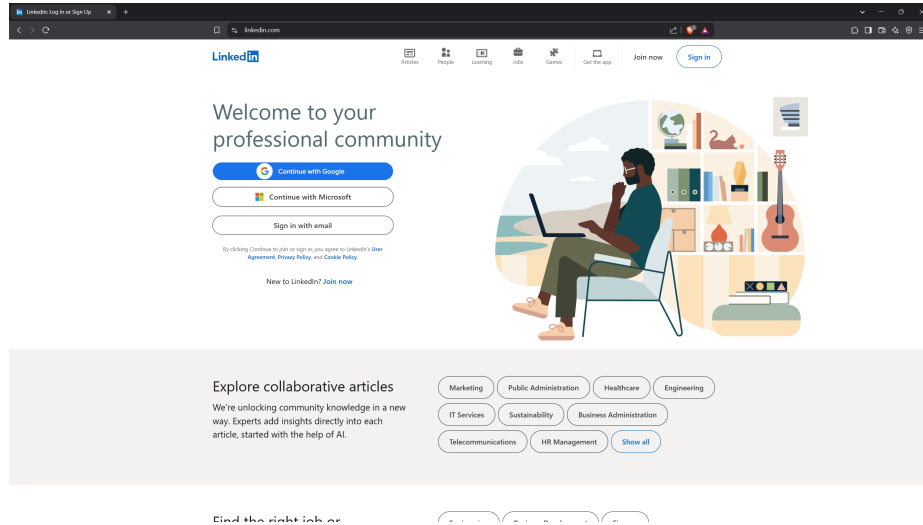


Fig. 1: Screenshot from the SEE Benchmark (image_id: 015): LinkedIn login page

Table 2: Annotation details for image_id: 015

Field	Content
image_id	015
reference_description	The screen shows LinkedIn’s login and sign-up page. At the top, the LinkedIn logo appears with the phrase “Welcome to your professional community.” Three login buttons are shown: “Continue with Google,” “Continue with Microsoft,” and “Sign in with email.” A short notice below refers to LinkedIn’s terms and privacy policy. Further down is a section titled “Explore collaborative articles” with category buttons like “Marketing,” “Public Administration,” and “Healthcare.” The top navigation bar lists items such as “Articles,” “People,” “Learning,” “Jobs,” and “Games.”
must_include	LinkedIn login screen; three login buttons: Google, Microsoft, Email

3.2 Model Selection

An important factor in model selection was long-term viability. Since the goal was to build a capable model that could be relied upon beyond the thesis period, models from smaller research teams were deliberately avoided, as they risked being discontinued or unsupported. Instead, the focus was on models backed by major institutions, such as Meta (LLaMA) and Alibaba (Qwen), due to their ongoing investment in open-source development [17,1]. This choice was also motivated by the rapid pace of LLM research: new models are released frequently, often outperforming earlier ones. By choosing a model family with strong continuity, any optimized prompt developed for a current version (e.g., LLaMA 3 or 4) would be more likely to transfer successfully to future versions in the same family, unlike prompts tuned for models with no clear successor.

Several model sizes were researched for feasibility, from (≤ 11 B parameters) for on-device solutions, up to (≤ 90 B parameters) server-run solutions. This allowed insight on how small a model should be to be feasible for this use-case.

Once the benchmark was complete, it enabled a structured comparison of model and prompt combinations. Two main strategies were considered: optimizing a prompt first and applying it across models to identify the strongest model, or selecting the best-performing model first and tailoring prompts for it. A hybrid strategy was ultimately adopted. Six prompt variants were tested across four models to identify high-performing combinations and ensure broader coverage. This allowed for more reliable model selection, as consistent underperformance across prompts revealed weaker models. These were excluded from further refinement, allowing more focused prompt iteration on the strongest candidate.

3.3 Evaluation Methods

Once the benchmark dataset was complete, the next question became: how can the quality of VLM-generated screen descriptions be measured? Unlike traditional tasks such as math problem solving, where correctness is binary, evaluating a screen description is inherently open-ended. The evaluation should assess the semantic quality of generated descriptions in relation to the human-written reference, while also ensuring that all essential elements are covered.

Initially, BERTScore was chosen to evaluate semantic similarity [29]. This metric compares each word in a generated description with words in the reference using contextual embeddings from a pre-trained language model. Unlike BLEU or ROUGE, BERTScore rewards paraphrasing and captures meaning at a contextual level.

To validate whether must-include elements were present in the output, a secondary check was implemented using a smaller LLM (Mistral-7B). This model was prompted to detect whether must-include elements, defined in the annotations, were semantically present.

While initially promising, BERTScore soon showed limitations: scores clustered within a narrow range (typically 0.70–0.75), and even visibly different outputs often received similar values.

Because of this, a more robust strategy was adopted: LLM-as-a-Judge evaluation using DeepEval, based on the G-Eval framework [15,8]. Research shows that G-Eval outperforms evaluation solutions like BERTScore in correlating with human judgements across language generation tasks [15]. This method uses a strong LLM, specifically GPT-4o, to score outputs based on natural language criteria. Each description was rated from 0 to 1 and accompanied by a brief justification. Two metrics were used:

Reference Coverage. This measures how well the generated description aligns with the annotated reference. A score of **0.7 or higher** was considered a pass. Evaluation steps included checking for Dutch language use, functional coverage, paraphrase tolerance, the presence of required reference elements, and the exclusion of hallucinated or purely visual details (Appendix A.1).

Must-Include Coverage. This assesses whether all must-include elements annotated in the data were present in the output, either literally or paraphrased. A score of **0.8 or higher** was required to pass. All must-includes were treated equally: if any were missing or unrecognisable, the score was lowered accordingly (Appendix A.2).

These thresholds and evaluation steps were established based on empirical tuning, using controlled test cases with known-good and known-bad outputs (Appendix A.3 & A.4). Each test case had an expected score range based on human judgment. When scores deviated from expectations, the evaluation criteria were revised: tightened if scores were too lenient, or relaxed if they were overly strict. This calibration process was critical to ensure that the right instructions were given to the LLM-as-a-judge to properly evaluate the performance of the model output compared to the reference description.

An overall pass was then granted if the model output passed on both the reference and must-include tests. The final implementation of the evaluation is shown in Figure 2.

3.4 Prompt Annotation Feedback Loop

Once the benchmark, chosen model and evaluation framework were in place, a feedback loop emerged between the prompt, the model outputs, and the reference annotations. This loop became central to improving the prompt, but most unexpectedly the reference descriptions.

As prompts became more refined, model outputs would occasionally include details that had been overlooked in the human-written reference descriptions.

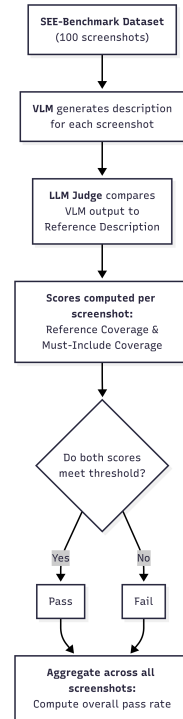


Fig. 2: Final SEE Benchmark Evaluation Process



Fig. 3: Feedback loop, as shown by DeepEval [8]

These cases were manually reviewed and, if they clearly improved the annotation, the insight was incorporated. This is in line with the structure proposed by DeepEval in Figure 3.

In incorporating these overlooked details, no direct model output was ever copied into the annotation dataset, to prevent possible prompt-overfitting. Instead, revisions were human-written and based on human judgment. For instance, if a model highlighted an important button or heading that was initially missed, this was added to the annotation dataset. If the output used phrasing that was clearer or more concise, it served as inspiration for a human-authored rewording that better captured the screen’s purpose.

Ultimately, there was a trade-off to be made between freezing the annotations early and preventing any chance of overfitting, or using the feedback loop to improve annotation quality, but creating the possibility of overfitting the prompt to the reference description. Given the nature of having an LLM-as-a-Judge semantically evaluate the output, rather than literally, the latter option was adopted. This provided a solution in which the model output would display a higher quality structure and detail to the eventual end-user, while remaining semantically similar in the eyes of the benchmark.

3.5 Prompt-Engineering Strategies

Prompt design played a central role in both the selection of the best-performing model as in overall model performance. The process was divided into two stages: identifying a model that performed well under different prompting conditions, and performing targeted prompt engineering on that model to maximise descriptive quality.

As described in Section 3.2, six prompt variants were used to evaluate baseline model performance. Full prompt texts are included in Appendix B.1–B.6:

- **Zero-Shot:** A minimal instruction prompt without structure or examples (Appendix B.1).

- **Role-Instructed:** A prompt framing the model as an assistant for blind users. (Appendix B.2).
- **Two-Shot:** A prompt including two example screen descriptions to guide output style and content (Appendix B.3).
- **Structured Output Prompt:** A prompt with explicitly formatted rules and description guidelines to promote consistency and completeness (Appendix B.4).
- **Dual-Pass Prompt:** A two-step instruction prompt where the model first analyses the layout, then generates a summary (Appendix B.5).
- **Saliency-Guided Prompt:** A prompt that makes the model focus on visually dominant and functionally central screen areas first (Appendix B.6).

These prompt types were chosen based on literature, where they were found to be good forms of early prompt engineering [4,22,14].

Prompt Refinement Stage. Once a best-performing model was established, prompt development proceeded in six iterative cycles. Prompts were revised based on feedback from the LLM-as-a-Judge system, which provided both numerical scores and brief natural language justifications. These justifications highlighted issues such as vague phrasing, missing must-include elements, or excessive verbosity. Each round of feedback informed the next prompt iteration, gradually improving coverage and clarity. The resulting prompt of this iterative refinement is called the Feedback-Prompt.

3.6 User Study

A user study was conducted with six blind government employees recruited via Visusrijk, all using screen readers as their primary assistive technology. The aim was to evaluate whether the VLM-generated screen descriptions would prove practically useful to the target audience. Each participant evaluated four screen types in fixed order: a JAWS installation screen, the Rijks-ICT dashboard, a PowerPoint presentation, and a KNGF donation form (Appendix D, images 006, 010, 049 and 084). These screens were generated descriptions on images from the annotation dataset. The screens were chosen to represent a realistic range of complex interfaces: a setup screen, a data-heavy dashboard, a presentation, and an online form.

To simulate realistic usage, each screen was introduced with a short real-world scenario (e.g., "You are about to install JAWS"). This mirrors how blind users typically know why they've opened a screen before exploring it, and therefore helps frame their expectations, resulting in a more relevant assessment of the VLM-generated descriptions.

The generated descriptions were played using Microsoft's Dutch text-to-speech (TTS) voice "Maarten", chosen for its similarity to voices used in popular screen readers, in order to reduce bias from pronunciation quality. Participants were explicitly instructed to focus on what was said, not how it was pronounced, as the TTS voice was not part of the evaluation. After each screen, participants rated the description using four questions: Q1 (clarity), Q2 (logical structure),

Q3 (appropriate description length), and Q4 (logical order). These ratings were recorded on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree). They also answered three open-ended questions about what stood out, whether they would ask follow-up questions, and whether anything was missing (Appendix C).

After completing all four screens, participants were asked three additional Likert questions: Q5 (usefulness in daily work), Q6 (increased perceived independence), and Q7 (sense of overview). These final questions were designed to reflect overall value and impact of the proposed solution as a whole.

Demographic data including age category, level of residual vision (e.g., light perception, partial vision, or total blindness), and whether the participant had any sight after age 6 were also collected to contextualize results. This was done to explore whether these factors influenced their preferences or understanding.

Responses were gathered through guided interviews and transcribed into a spreadsheet (Appendix C).

4 Experiments and Results

4.1 Evaluation of Small-Scale Models

Model scale proved to be a key factor in the quality of screen descriptions. Smaller models were briefly evaluated to determine feasibility, but their performance was so poor that formal metrics were deemed unnecessary.

Models such as LLaVA 7B and 11B, LLaVA-LLaMA3 8B, and Qwen 2.5 VL at 3B and 7B parameters were unable to produce coherent outputs in Dutch. Many failed to respond in the correct language, and those that did often returned irrelevant or hallucinated content. For example, a screenshot of a Dutch government website was repeatedly described as a COVID-19 information page despite no reference to COVID on the screen. These models appeared to overgeneralize based on superficial visual patterns and lacked the capability to interpret the actual interface. This phase was sufficient to establish that models at this scale are not viable for screen description tasks.

The next phase focused on models in the 72B to 90B range, selected primarily based on their long-term viability and institutional backing. Two models were chosen as main candidates for evaluation: LLaMA 3.2 Vision-Instruct at 90B parameters, and Qwen 2.5 VL at 72B parameters. Both models produced coherent Dutch descriptions with sufficient structure and detail to warrant full benchmark evaluation.

While testing was underway, Meta released LLaMA 4, which introduced a new architecture using a Mixture-of-Experts (MoE) approach. The 17B-128E Maverick and 17B-16E Scout variants became available, capable of selectively activating different expert pathways when needed. Although these models have only 17 billion parameters nominally, the MoE design effectively provides a much larger capacity. As a result, the 17B-128E variants visibly rivalled the 72B to 90B scale models in output quality and were thus also included in the full benchmark evaluation (Table 3).

Table 3: Models included for benchmark evaluation

Model	Parameters	Institution
LLaMA 3.2 Vision-Instruct	90B	Meta
Qwen 2.5 VL	72B	Alibaba
LLaMA 4 Maverick	17B-128E (MoE)	Meta
LLaMA 4 Scout	17B-16E (MoE)	Meta

4.2 Evaluation of Larger Models using SEE

During benchmarking, it became evident that Qwen 2.5 VL (72B) struggled with consistent Dutch language generation. Although outputs were mostly in Dutch, the model frequently generated unrelated characters or words from other languages such as Korean, Chinese, or Swedish within its descriptions. This erratic output made the model unfit for the use-case and was therefore eliminated from the pool.

LLaMA 3.2 Vision-Instruct (90B) performed better than Qwen, producing mostly coherent Dutch outputs, but sometimes generating fully English descriptions. While these English outputs were structurally sound, the inconsistency in language was problematic. This issue might have been addressable with further prompt engineering or tuning. However, since the LLaMA 4 variants were able to produce consistent Dutch descriptions on only a simple zero-shot prompt (Appendix B.1), LLaMA 3.2 was eliminated from further testing.

The two remaining models, LLaMA 4 Maverick and LLaMA 4 Scout, were evaluated on the full SEE Benchmark using six prompting strategies: Zero-Shot, Role-Instructed, Two-Shot, Dual-Pass, Structured, and Saliency-Guided (Appendix B.1–B.6). For each prompt, the average *Reference Coverage* score and *Must-Include Coverage* score were computed, alongside the percentage of outputs that were deemed as a pass on these metrics (score of ≥ 0.7 and ≥ 0.8 respectively). If a model output passed both thresholds, it was deemed an Overall Pass.

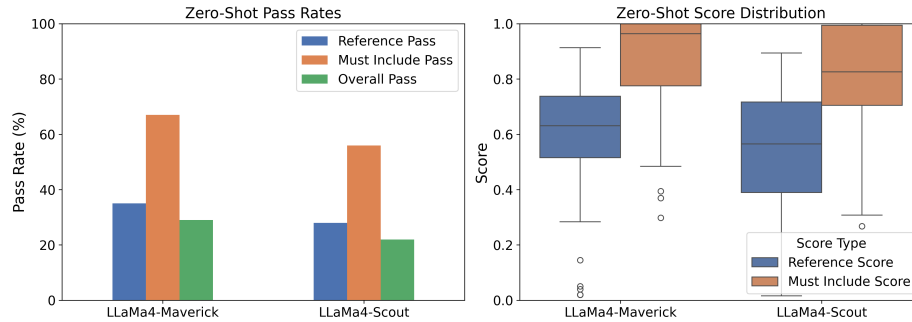


Fig. 4: Zero-Shot prompt evaluation of Maverick and Scout

In the Zero-Shot condition (Figure 4), Maverick significantly outperformed Scout on both Reference Score ($p = 0.0362$) and Must-Include Score ($p =$

0.0207). However, no significant differences were found in pass rates (all $p > 0.14$). This suggests that Maverick performed more reliably in minimal instruction settings, yielding higher average scores, but not to the extent that it significantly increased the proportion of outputs passing the thresholds.

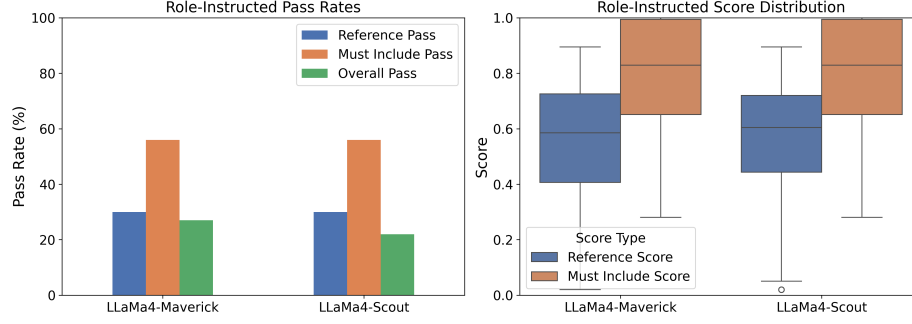


Fig. 5: Role-Instructed prompt evaluation of Maverick and Scout

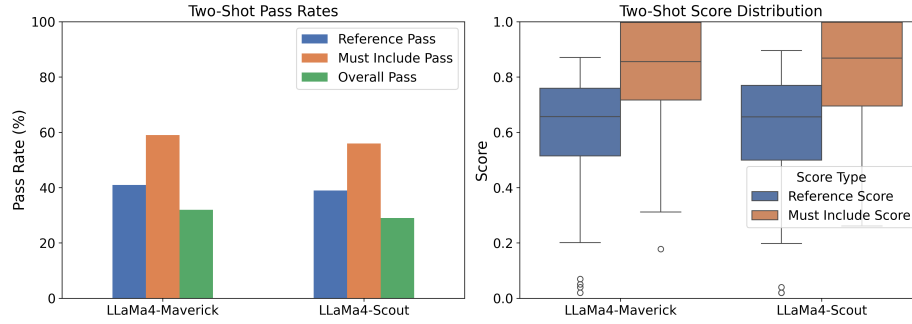


Fig. 6: Two-Shot prompt evaluation of Maverick and Scout

In the Role-Instructed and Two-Shot conditions (Figures 5 and 6), no statistically significant differences were observed between the models in either scores or pass rates (all $p > 0.6$).

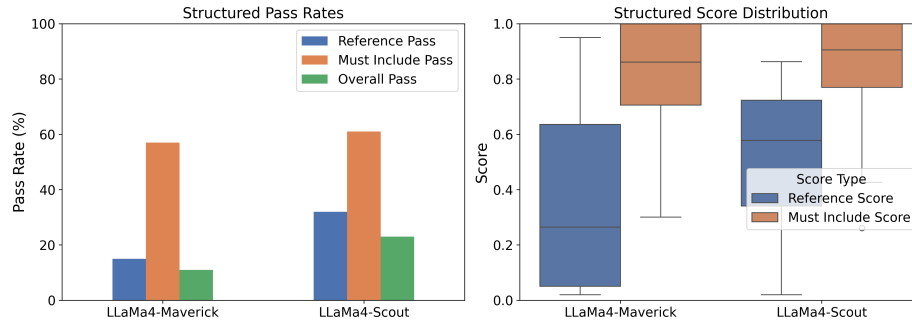


Fig. 7: Structured Output prompt evaluation of Maverick and Scout

The Structured Output prompt showed a statistically significant advantage for Scout in Reference Score ($p = 0.0001$) and Overall Pass Rate ($p = 0.0384$), though not in Must-Include Score (Figure 7). This indicates that while Scout occasionally produced more complete descriptions under structured guidance, it was not consistently better at including the critical elements defined in the benchmark. Furthermore, no significant differences were observed for the Dual-Pass or Saliency-Guided prompts (Figures 8 & 9).

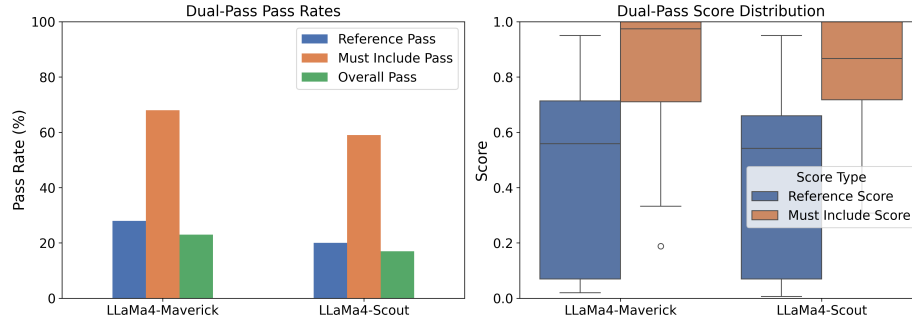


Fig. 8: Dual-Pass prompt evaluation of Maverick and Scout

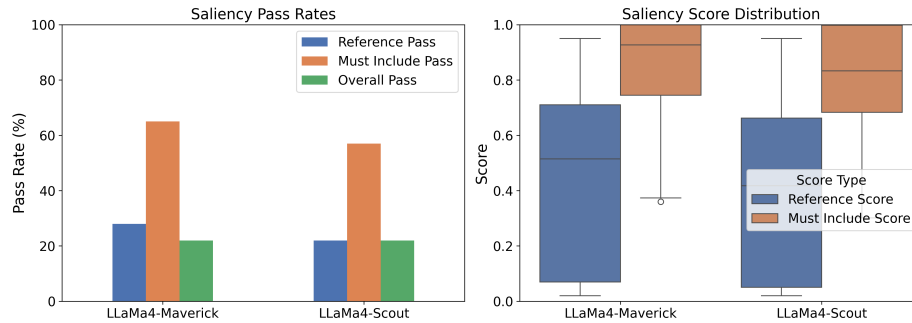


Fig. 9: Saliency-Guided prompt evaluation of Maverick and Scout

Figure 10 shows that Maverick performed better than Scout on most prompts. The Two-Shot prompt achieved the highest overall pass rate across both models. However, there was no significant difference between Maverick’s performance in the Two-Shot and Zero-Shot conditions, suggesting that minimal prompting may already yield strong results for this model.

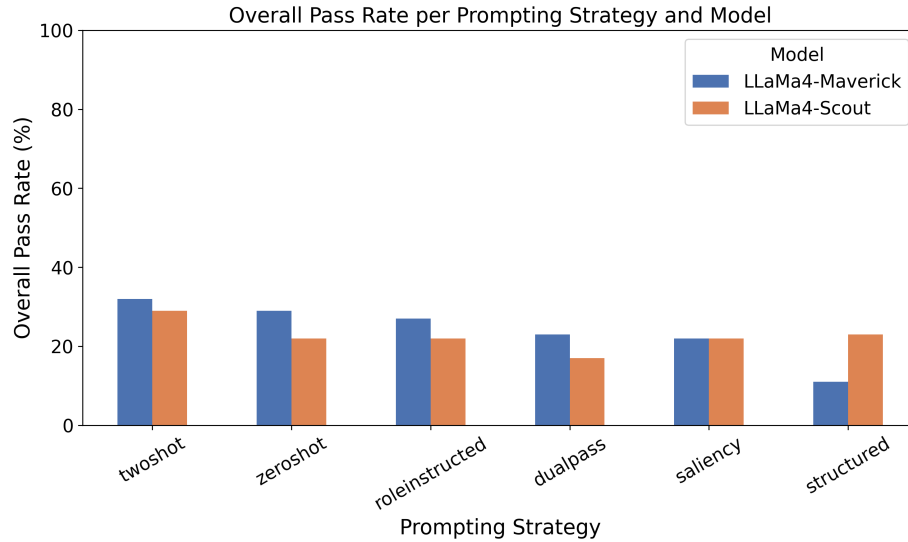


Fig. 10: Average Overall Pass Rate per Prompting Strategy and Model

Conclusion. While Scout significantly outperformed Maverick under the Structured Output prompt, this strategy did not yield high overall pass rates. In contrast, Maverick significantly outperformed Scout in the Zero-Shot condition, which was the second-best scoring prompt overall. Since Scout never significantly outperformed Maverick on any high-performing prompt, and Maverick showed stronger results with minimal instructions, it was selected for further prompt refinement in the next section.

4.3 Feedback-based Prompt

After selecting LLaMA 4 Maverick as the best-performing model, a final prompt refinement stage was conducted. This involved six iterations of tuning based on feedback from the LLM-as-a-Judge system, as described in Section 3.5. The resulting prompt, referred to as the **Feedback Prompt**, was designed to address often-occurring weaknesses identified during evaluation (Appendix B.7). The Feedback Prompt was evaluated against the previously strongest prompt (Two-Shot), again using the SEE Benchmark.

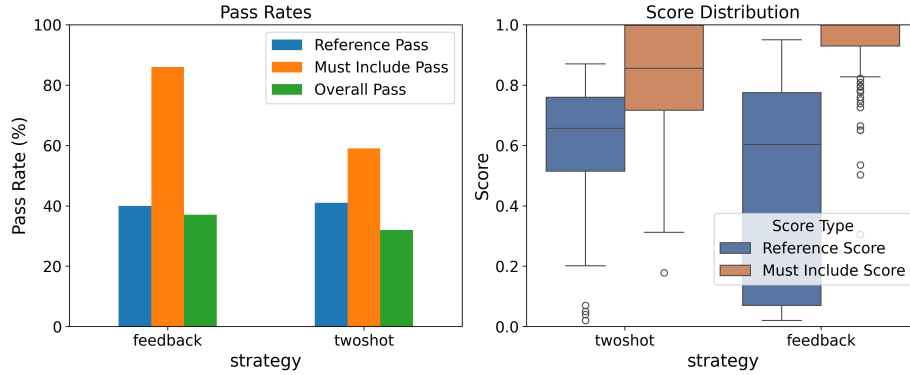


Fig. 11: Comparison of Feedback Prompt and Two-Shot Prompt on Maverick.

Although the Overall and Reference Pass did not increase significantly, the Feedback Prompt led to a significant improvement in Must-Include coverage (Figure 11). Both the average Must-Include Score and the number of outputs that passed the Must-Include threshold increased substantially ($p = 0.0000$). This suggests that the added structure and constraints helped the model focus more reliably on critical screen content, resulting in more consistent inclusion of essential elements.

At the same time, the average Reference Score was significantly higher under the Two-Shot prompt ($p = 0.0009$), indicating that its outputs more closely matched the original reference descriptions in style and phrasing. However, no difference was observed in Reference Pass Rate ($p = 1.0000$), meaning that both prompts achieved a similar level of alignment with the benchmark threshold.

These results show that by refining prompts through iterative feedback, models can be encouraged to include critical elements more consistently. At the same time, this refinement introduced a trade-off: descriptions began to diverge more from the original reference, resulting in a lower Reference Score. This suggests that incorporating model feedback too directly can shift the model’s focus toward satisfying specific checklist-like expectations, rather than matching the phrasing and structure of the original annotation.

As the Feedback Prompt performed significantly better in Must-Include coverage, without performing significantly worse on overall or reference-based metrics, this prompt was selected for use in the user study described in the next section.

4.4 User Study with Blind Users

Participants responded very positively to the VLM-generated screen descriptions. Ratings for clarity (Q1), structure (Q2), length (Q3), and order (Q4) were consistently high across most screen types. In particular, the PowerPoint, donation form, and installation screen were all seen as clear, well-structured, concise, and logically ordered, with average scores on each aspect exceeding 4 (Figure 12).

Only the dashboard received somewhat lower scores across all four aspects, suggesting that its description may have contained more information than participants preferred to receive in one go. While the output was still considered useful, several participants indicated a desire for a more gradual structure: starting with a global overview, followed by the option to explore specific components in more detail. This suggests that while screens with less dense visual content can be effectively conveyed in a single pass, complex, data-heavy dashboards may benefit from a layered approach in which users first receive a global overview and then have the ability to ask further questions.

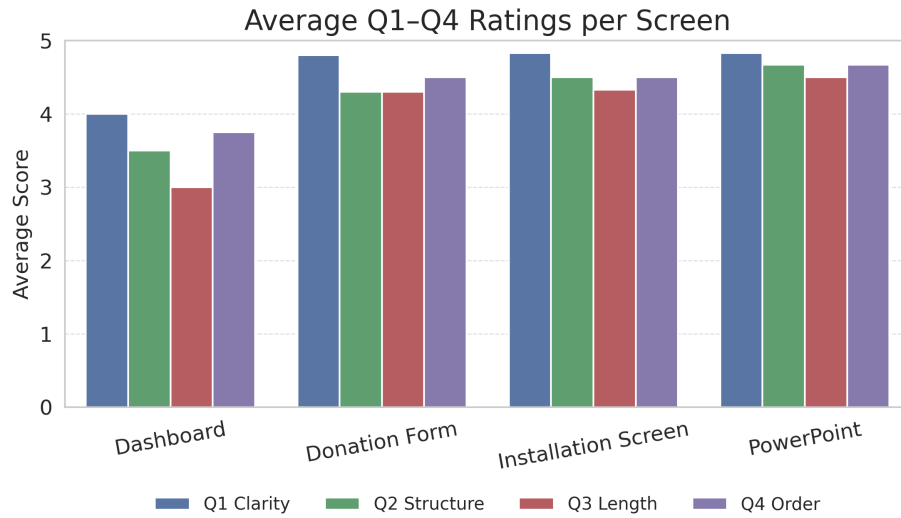


Fig. 12: Average scores on Q1–Q4 per screen

In addition to per-screen evaluations, participants answered three final questions about the overall usefulness of the system in their daily work (Q5), whether it increased their sense of independence (Q6), and whether it provided a better overview of the screen than their screen reader typically offers (Q7). All of these received near-perfect scores: 5.0, 4.83, and 4.83 respectively, indicating strong perceived value. Participants described the experience as “helpful” and “complete,” with one stating it was like “having a second pair of eyes,” and another comparing it to “having someone look over your shoulder and describe the screen to you.”

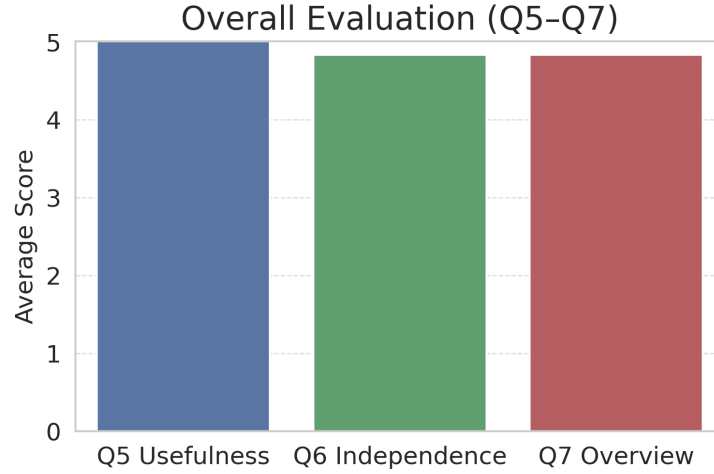


Fig. 13: Average scores on Q5–Q7 per screen

Finally, users shared preferences about how information should be presented. Several wanted the purpose of the screen to be stated at the beginning rather than at the end. Some commented negatively on vague expressions such as “seems active,” preferring more assertive and unambiguous phrasing. A few participants also noted that the system occasionally made interpretive statements (e.g., inferring the “goal” of a screen), and they emphasized a preference for factual, non-interpretive descriptions.

5 Conclusion

This thesis introduced the SEE Benchmark to systematically evaluate vision-language models (VLMs) for desktop screen descriptions aimed at visually impaired users. The evaluation showed that smaller-scale VLMs were incapable of reliably producing coherent Dutch descriptions, answering **RQ4** by identifying that larger-scale models, notably LLaMA 4 Maverick, were required for consistent performance.

Prompt design significantly impacted the description quality of the models, with some prompts consistently performing better than others across both models, thus directly addressing **RQ3**.

Automated evaluation of VLM-generated descriptions (**RQ2**) was successfully implemented through an LLM-as-a-Judge pipeline, providing reproducible, objective measures aligned with annotated reference descriptions and essential element requirements.

Finally, a user study demonstrated that VLM-generated descriptions substantially improved desktop interface understanding compared to traditional screen readers. Participants unanimously reported increased independence and improved screen overview, clearly affirming the practical benefit and answering

RQ1. They also highlighted the preference for factual descriptions and interactive, layered exploration for complex interfaces, suggesting valuable directions for future work.

Although this research focused on Dutch screens and involved a limited participant group, the methodology provides a scalable and generalisable foundation for future development. Future work should expand the annotation dataset to include additional languages, refine and scale the annotations to optimise scoring reliability, and conduct a larger-scale user study to further validate the system's effectiveness.

A Appendix A: Evaluation Test Cases

A.1 Reference Coverage Evaluation Steps

- De beschrijving moet volledig in het Nederlands zijn. Engelse termen zijn alleen toegestaan als ze letterlijk ook in de referentie staan. Als er Engelse termen voorkomen die niet in de referentie staan, geef dan een lage score (onder 0.3).
- Controleer of alle belangrijke en functioneel relevante elementen uit de referentie aanwezig zijn in de modeloutput, ook als ze net anders genoemd worden, zolang de essentie van het functioneel relevante element maar overkomt. Als er iets ontbreekt, trek dan punten af.
- Parafrasering is toegestaan, zolang betekenis en inhoud behouden blijven. Als de betekenis verandert of vager wordt, trek dan punten af.
- Als er elementen genoemd worden die niet in de referentie staan (zoals extra knoppen of functionaliteiten), trek dan punten af.
- Als de modeloutput visuele of stilistische details bevat die niet in de referentie staan én geen functioneel doel dienen, trek dan punten af.
- Als de modeloutput veel langer is dan nodig of onnodige omschrijvingen bevat, trek dan punten af.

A.2 Must-Include Coverage Evaluation Steps

- Controleer voor elk verplicht element of het in de output staat. Synoniemen of parafraseringen tellen even zwaar mee.
- Als alles aanwezig is, telt dit sterk mee in de beoordeling.

This appendix contains real test cases used to calibrate the evaluation steps for the SEE Benchmark. These examples were used to test whether the LLM-as-a-Judge system (GPT-4o via DeepEval) scored outputs in expected ranges based on their semantic quality and completeness.

A.3 Reference Coverage Tuning

All five outputs below were evaluated against the same reference description:

Reference: Op de pagina staat bovenaan een donkerblauwe balk met het logo van Microsoft Teams en een knop 'Sign out'. In het midden staat de titel "*Join a meeting*". Daaronder zijn twee velden: '*Meeting ID*' en '*Passcode*'. Onderaan staat een knop '*Join now*' en daarnaast een link '*Learn more*'.

Case	Model Output	Expected Score
1	De webpagina toont bovenaan een donkerblauwe balk met het Microsoft Teams-logo en een knop 'Sign out'. In het midden staat de tekst "Join a meeting" met daaronder twee velden: 'Meeting ID' en 'Passcode'. Onderaan staan de knoppen 'Join now' en 'Learn more'.	0.95–1.00
2	Op de pagina staat een blauwe balk met logo en 'Sign out'. Centraal staat "Join a meeting" in grote letters, gevolgd door velden 'Meeting ID', 'Passcode'. Onderaan 'Join now' (blauwe knop) en 'Learn more'. Layout is responsief en modern.	0.68–0.74
3	Bovenaan staat het Teams-logo en 'Sign out'. Middenin staat "Join a meeting" met drie velden: 'Meeting ID', 'Passcode', en 'Naam'. Onderaan: 'Join now' en 'Learn more'.	0.58–0.65
4	Bovenaan de pagina staat een donkerblauwe balk met links het Microsoft Teams-logo en rechts de knop 'Sign out'. In het midden van het scherm staat in grote letters "Join a meeting"... At the top of the screen is a dark blue bar...	0.00
5	De pagina toont bovenaan een blauwe balk met het logo en een knop 'Afmelden'. In het midden staat 'Join a meeting' en velden 'Meeting ID', 'Passcode'. Onderaan de pagina: knoppen 'Join now' en 'Meer info'.	0.00

A.4 Must-Include Coverage: Tuning

For the following five cases, the required must-include elements were: "zoekbalk", "knop 'Inloggen'", "menu"

Case	Model Output	Expected Score
1	Bovenaan de pagina staat een zoekbalk, rechts daarvan de knop 'Inloggen', en onderaan een menu met verschillende opties.	1.00
2	Helemaal bovenin is een veld om te zoeken, daarnaast vind je de login-knop. Onderaan staan de menu-opties.	1.00
3	De pagina bevat een zoekbalk, een knop 'Inloggen', een afbeelding en een helpknop.	0.67
4	Er is een inlogknop rechtsboven.	0.33
5	Deze pagina toont een afbeelding van een landschap met een weerbericht erboven.	0.00

B Appendix B: Prompt Variants Used in Model Comparison

This appendix lists the three prompt variants used in the benchmark evaluation of LLaMa 3.2, QWEN 2.5VL, LLaMA 4 Maverick and Scout, as described in Section 4.2.

B.1 Zero-Shot Prompt

Beschrijf dit scherm aan een blinde gebruiker.

B.2 Role-Instructed Prompt

Je bent een assistentie-agent voor blinde gebruikers. Beschrijf dit scherm op een duidelijke en beknopte manier. Leg uit wat het doel van het scherm is, benoem de visuele structuur en noem belangrijke elementen zoals koppen, knoppen en tekstvelden. Doe dit alsof je het aan iemand samenvat die het scherm niet kan zien.

B.3 Two-Shot (Few-Shot Contextual) Prompt

Prompt:

Beschrijf dit scherm aan een blinde gebruiker.

Voorbeeld 1:

Afbeelding: Een webpagina van de Belastingdienst.

Beschrijving: Deze pagina toont vier blokken met informatie over belastingaangifte, inclusief deadlines, loginmogelijkheden en belangrijke thema's zoals toeslagen en erfbelasting. Bovenaan staan navigatie-opties, een zoekveld en een inlogknop.

Voorbeeld 2:

Afbeelding: Een inlogscherm van een bankapplicatie.

Beschrijving: Dit scherm vraagt om een gebruikersnaam en wachtwoord. Onderaan staat een knop 'Inloggen' en een link voor 'Wachtwoord vergeten'. Bovenaan staat het banklogo en de naam van de app.

Jouw taak:

Beschrijf dit scherm kort en duidelijk voor een blinde gebruiker.

Geef aan wat het doel van de pagina is en welke functies of onderdelen belangrijk zijn om te begrijpen of gebruiken.

Laat geen belangrijke onderdelen weg. Zorg dat de gebruiker weet wat hij op deze pagina kan doen.

B.4 Structured Output Prompt

Beschrijf deze schermafbeelding in één feitelijke alinea (5–7 zinnen) voor een blinde gebruiker.

Begin met het type scherm en, indien zichtbaar, de naam van de applicatie of website. Noem vervolgens het doel van het scherm en beschrijf alle relevante visuele elementen: titels, knoppen, tabbladen, invoervelden of andere interactieve onderdelen. Gebruik letterlijk zichtbare labels of namen.

Geef bij meerdere gelijkaardige elementen minstens drie voorbeelden (zoals: "A", "B" en "C") en vermeld voortgangstatus of foutmeldingen indien aanwezig.

Vermijd decoratieve details zoals kleuren of stijl. Gebruik neutrale en feitelijke taal, zonder opsommingen of bullet points.

B.5 Dual-Pass Prompt

Stap 1 – Analyseer het scherm:

- Wat voor soort scherm is dit?
- Wat is het hoofdonderwerp of de functie?
- Welke zichtbare elementen zijn belangrijk voor gebruik?

Stap 2 – Genereer de beschrijving:

Gebruik je analyse om het scherm in één feitelijke alinea (5–7 zinnen) te beschrijven voor een blinde gebruiker. Begin met het type en doel van het scherm, en benoem vervolgens alle functioneel relevante onderdelen, inclusief knoppen, titels en invoervelden. Gebruik letterlijk zichtbare labels waar mogelijk. Vermijd decoratieve of irrelevante details.

B.6 Saliency-Guided Prompt

Beschrijf deze schermafbeelding voor een blinde gebruiker, met nadruk op de belangrijkste en functioneel relevante onderdelen. Begin met het type scherm en de applicatie of website (indien zichtbaar). Beschrijf vervolgens het hoofddoel van het scherm en de belangrijkste elementen die een gebruiker nodig heeft om ermee te werken: knoppen, knoppen, formulieren, foutmeldingen of statusinformatie. Vermijd irrelevante of decoratieve details. Noem alleen elementen die bijdragen aan begrip of interactie met het scherm. Houd de beschrijving compact, neutraal en volledig in het Nederlands.

B.7 Feedback Prompt (Final Version)

Stel je voor dat je dit scherm beschrijft aan een blinde gebruiker die de pagina niet kan zien. Je beschrijving moet hen in staat stellen de pagina te begrijpen en te gebruiken. Volg deze richtlijnen:

1. Beschrijf het scherm volledig in het Nederlands.
2. Neem alle functioneel relevante elementen uit de referentie op, zoals specifieke tekst, titels, data, numerieke waarden, en posities van elementen (bijvoorbeeld links, rechts, bovenaan, onderaan).
3. Voeg geen details toe die niet in de referentie staan, zoals extra knoppen, afbeeldingen of speculatieve functionaliteiten.
4. Gebruik alleen Engelse of andere niet-Nederlandse termen als ze letterlijk op de pagina staan en in de referentie voorkomen.
5. Behoud de exacte betekenis, inhoud en structuur bij parafrasering; vermijd vaagheid of wijzigingen die de functionaliteit beïnvloeden.
6. Beschrijf de exacte positionering en hiërarchie van elementen, zoals links, knoppen en formulieren, zoals aangegeven in de referentie.
7. Geef het doel van de pagina en de belangrijkste functies duidelijk aan.
8. Zorg dat de beschrijving kort, duidelijk en functioneel bruikbaar is voor een blinde gebruiker.

Voorbeeld 1:

Afbeelding: Een webpagina van de Belastingdienst.

Beschrijving: Deze pagina toont vier informatieblokken over belastingaangifte met specifieke deadlines, zoals "Aangifte inkomstenbelasting doen" tot 1 mei, en links naar onderwerpen zoals toeslagen en erfbelasting. Bovenaan is een navigatiebalk met opties zoals "Home", "Menu", een zoekbalk en een inlogknop.

Voorbeeld 2:

Afbeelding: Een inlogscherf van een bankapplicatie.

Beschrijving: Dit scherm vraagt om een gebruikersnaam en wachtwoord in invoervelden. Onderaan staan een knop "Inloggen" en een link "Wachtwoord vergeten". Bovenaan is het banklogo en de naam van de applicatie zichtbaar.

Jouw taak:

Beschrijf het scherm volgens de bovenstaande richtlijnen.

C Appendix C: User Study Questions

Schermen: De studie omvatte vier schermen met een korte inleiding voor context, gevolgd door een AI-gegenereerde beschrijving:

1. **JAWS:** U start uw computer op een nieuwe werkplek, opent het JAWS-installatiebestand en vraagt een beschrijving van de voortgang.
2. **Rijks ICT-dashboard:** U navigeert via rijksictdashboard.nl naar een pagina met grafieken over ICT-projecten binnen de rijksoverheid.
3. **PowerPoint-presentatie Beeldherkenner:** U opent een PowerPoint over een hulpmiddel voor blinde medewerkers en vraagt een beschrijving.
4. **KNGF Donatieformulier:** U klikt op "Ik wil doneren" op de KNGF Geleidehonden-website en vraagt een beschrijving.

Vragen per scherm: Deelnemers beantwoordden per scherm (1 = helemaal oneens, 5 = helemaal eens) met een korte toelichting:

1. De beschrijving gaf een duidelijk beeld van het scherm.
2. De beschrijving was logisch, gestructureerd en prettig.
3. De lengte van de beschrijving was passend.
4. De volgorde van informatie was logisch.

Open vragen:

- Wat viel op aan de beschrijving (verwarrend, overbodig, goed)?
- Zou u een navraag stellen? Zo ja, welke?
- Heeft u iets gemist?

Algemene vragen: Deelnemers beantwoordden (1 = helemaal oneens, 5 = helemaal eens):

1. Ik zou deze beschrijvingen nuttig vinden in mijn werk.
2. Ik zou me zelfstandiger voelen met deze beschrijvingen.
3. Deze beschrijvingen bieden meer overzicht dan mijn screenreader.

Demografische en contextvragen:

- **Leeftijdscategorie:** <30 30–50 >50
- **Zicht:** Blind Slechtziend
- **Zicht na zesde levensjaar?**

D Appendix D: SEE Benchmark Source Code and Annotation Dataset

The full source code and annotation dataset are available at <https://github.com/CedrickMakhlouf/SEE-Benchmark>.

References

1. Alibaba Cloud Qwen Team: Qwen: General-purpose open-source llms by alibaba cloud. <https://github.com/QwenLM> (2025), accessed: 2025-06-29
2. Apple Inc.: Voiceover – accessibility features. <https://www.apple.com/accessibility/> (2025), accessed: 2025-06-29
3. Be My Eyes: Be my eyes for windows: Ai-powered desktop accessibility. <https://www.bemyeyes.com/be-my-eyes-for-windows/> (2025), accessed: 2025-06-29
4. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners (2020), <https://arxiv.org/abs/2005.14165>
5. Bucciarelli, D., Moratelli, N., Cornia, M., Baraldi, L., Cucchiara, R.: Personalizing multimodal large language models for image captioning: An experimental analysis (2024), <https://arxiv.org/abs/2412.03665>
6. Chemnad, K., Othman, A.: Digital accessibility in the era of artificial intelligence—bibliometric analysis and systematic review. *Frontiers in Artificial Intelligence* **7**, 1349668 (2024). <https://doi.org/10.3389/frai.2024.1349668>, <https://www.frontiersin.org/articles/10.3389/frai.2024.1349668>
7. Chen, Z., Liu, Z., Wang, K., Wang, K., Lian, S.: A large vision-language model based environment perception system for visually impaired people. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). p. 221–228. IEEE (Oct 2024). <https://doi.org/10.1109/iros58592.2024.10801813>, <http://dx.doi.org/10.1109/IROS58592.2024.10801813>
8. Confident AI Inc.: Deepeval documentation: The open-source llm evaluation framework. <https://deepeval.com/docs/getting-started> (2025), accessed: 2025-06-29
9. Envision Technologies: Envision glasses: Ai-powered smartglasses for blind and low vision users. <https://www.letsenvision.com/glasses> (2025), accessed: 2025-06-29
10. Freedom Scientific: New and improved features in jaws. <https://www.freedomscientific.com/training/jaws/new-and-improved-features/> (2024), accessed: 2025-06-29
11. Fujitake, M.: Layoutllm: Large language model instruction tuning for visually rich document understanding (2024), <https://arxiv.org/abs/2403.14252>
12. Gooda Sahib-Kaudeer, N.: Investigating the information seeking behaviour of blind searchers on the web (07 2011). <https://doi.org/10.14236/ewic/HCI2011.9>

13. Google LLC: Gemini ai: Updates to android and pixel at made by google 2024. <https://blog.google/products/gemini/made-by-google-gemini-ai-updates/> (2024), accessed: 2025-06-29
14. Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., Sabharwal, A.: Decomposed prompting: A modular approach for solving complex tasks (2023), <https://arxiv.org/abs/2210.02406>
15. Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C.: G-eval: Nlg evaluation using gpt-4 with better human alignment (2023), <https://arxiv.org/abs/2303.16634>
16. Lovells, H.: The european accessibility act: What should financial services firms be focusing on as the june 2025 compliance deadline approaches? <https://www.lexology.com/library/detail.aspx?g=780313fc-3abc-40fc-9123-3d05ba5aa098> (2025), accessed: 2025-06-29
17. Meta Platforms Inc.: Llama 3: Open-source large language models by meta. <https://github.com/meta-llama> (2025), accessed: 2025-06-29
18. Meta Platforms Inc.: Meta ai-powered ray-ban smart glasses. <https://about.fb.com/news/2024/09/ray-ban-meta-glasses-new-ai-features-and-partner-integrations/> (2025), accessed: 2025-06-29
19. Microsoft: Seeing ai: Talking camera for the blind. <https://www.seeingai.com/> (2025)
20. Naayini, P., Myakala, P.K., Bura, C., Jonnalagadda, A.K., Kamatala, S.: Ai-powered assistive technologies for visual impairment (2025), <https://arxiv.org/abs/2503.15494>
21. OrCam Technologies: Orcam myeye 3 pro: Wearable assistive device for the visually impaired. <https://www.orcam.com/en/myeye> (2025)
22. Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., Dulepet, P.S., Vidyadhara, S., Ki, D., Agrawal, S., Pham, C., Kroiz, G., Li, F., Tao, H., Srivastava, A., Costa, H.D., Gupta, S., Rogers, M.L., Goncarenco, I., Sarli, G., Galynker, I., Peskoff, D., Carpuat, M., White, J., Anadkat, S., Hoyle, A., Resnik, P.: The prompt report: A systematic survey of prompt engineering techniques (2025), <https://arxiv.org/abs/2406.06608>
23. W3C: Web content accessibility guidelines (wcag). <https://www.w3.org/WAI/standards-guidelines/wcag/> (2025), accessed: 2025-06-29
24. Walle, H., De Runz, C., Serres, B., Venturini, G.: A survey on recent advances in ai and vision-based methods for helping and guiding visually impaired people. *Applied Sciences* **12**(5), 2308 (2022). <https://doi.org/10.3390/app12052308>, <https://www.mdpi.com/2076-3417/12/5/2308>
25. WebAIM: Screen Reader User Survey 9. <https://webaim.org/projects/screenreadersurvey9/> (2021)
26. WebAIM: Screen reader user survey #10 results. <https://webaim.org/projects/screenreadersurvey10/> (2024)
27. WebAIM: The webaim million: An annual accessibility analysis of the top 1,000,000 home pages. <https://webaim.org/projects/million/> (2024)
28. Xu, P., Ding, Y., Fan, W.: Chartadapter: Large vision-language model for chart summarization (2024), <https://arxiv.org/abs/2412.20715>
29. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert (2020), <https://arxiv.org/abs/1904.09675>
30. Zhang, X., de Greef, L., Swearngin, A., White, S., Murray, K., Yu, L., Shan, Q., Nichols, J., Wu, J., Fleizach, C., Everitt, A., Bigham, J.P.: Screen recognition: Creating accessibility metadata for mobile applications from pixels. In:

Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. ACM (2021). <https://doi.org/10.1145/3411764.3445186>, <https://doi.org/10.1145/3411764.3445186>