An Experience Report on Leveraging LLMs for GUI Generation: Automating Coding to Prioritise Creativity ⋆

Giovanna Broccia[1,*], Alessandro Borselli[2], Maria Rosaria Cefaloni[2], Franco Delcorno[2] and Alessio Ferrari[1,3]

[1]CNR-ISTI, Pisa, Italy
[2]Trenord, Milan, Italy
[3]University College Dublin, Dublin, Ireland

Abstract

The design of graphical user interfaces (GUIs) is a complex and time-consuming process that begins with identifying user roles and gathering requirements through interviews, surveys, or workshops. Designers then create low-fidelity sketches or digital wireframes, organising information into logical sections and selecting visual elements to enhance usability. This iterative process often demands extensive refinement based on stakeholder feedback, making mockup creation—especially for interactive prototypes—a time-consuming task. In particular, the mockup development process often entails spending significant effort on clerical activities, such as programming and debugging tasks, rather than concentrating on creativity, human interaction, and quick feedback cycles with stakeholders.

This paper investigates whether large language models (LLMs) can assist GUI designers in streamlining the design process—reducing time and effort while maintaining design quality—enabling them to focus on the human aspects of creativity and user interaction by offloading technical programming tasks to the machine. We document our experience designing a dashboard for predictive maintenance in railways, illustrating how LLMs can support key tasks such as requirement analysis, information organisation, and mockup generation and refinement. We discuss insights and lessons learned, including the importance of clear requirements, the impact of LLM choice, and the benefits of iterative refinement in achieving stakeholder alignment. Our study shows that LLMs can support the GUI design process by automating specific tasks, thereby reducing design effort and enhancing the overall quality and satisfaction of the final product.

Keywords

GUI Design, LLMs, ChatGPT, DeepSeek, Requirement Engineering

1. Introduction

The development of graphical user interfaces (GUIs) is a dynamic and user-centred process that thrives on continuous collaboration with end-users and stakeholders to create intuitive and effective designs [1]. When developing GUIs, it is particularly important to elicit the needs, behaviours, and goals of users, ensuring the final product aligns seamlessly with their expectations. The process begins by identifying diverse user roles and personas, which helps in tailoring the interface to meet the specific needs of different user groups. Methods such as interviews, surveys, and interactive workshops are employed to gather requirements, fostering direct engagement with stakeholders [2]. This participatory approach not only uncovers their goals and pain points but also empowers users to contribute to the design process,

In: A. Hess, A. Susi, E. C. Groen, M. Ruiz, M. Abbas, F. B. Aydemir, M. Daneva, R. Guizzardi, J. Gulden, A. Herrmann, J. Horkoff, S. Kopczyńska, P. Mennig, M. Oriol Hilari, E. Paja, A. Perini, A. Rachmann, K. Schneider, L. Semini, P. Spoletini, A. Vogelsang. Joint Proceedings of REFSQ-2025 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2025. Barcelona, Spain, April 7, 2025.

⋆Research supported by the EU Project CODECS GA 101060179. This study was carried out within the MOST – Sustainable Mobility National Research Center and received funding from the European Union NextGenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4, COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

*Corresponding author.
giovanna.broccia@isti.cnr.it (G. Broccia); alessandro.borselli@trenord.it (A. Borselli); mariarosaria.cefaloni@trenord.it (M. R. Cefaloni); franco.delcorno@trenord.it (F. Delcorno); alessio.ferrari@ucd.ie (A. Ferrari)
0000-0002-4737-5761 (G. Broccia); 0000-0002-0636-5663 (A. Ferrari)

enhancing ownership and satisfaction [3]. Once requirements are elicited, they are systematically analysed and structured to prioritise features. To this end, designers typically organise information by grouping related data and functionalities into logical sections, defining an information hierarchy to highlight critical content, and selecting appropriate visual elements to enhance usability, consistency, and simplicity. This structured information serves as the foundation for interface layout design, which often starts with low-fidelity sketches or digital wireframes to visualise initial concepts [4].

Refining these mockups based on stakeholder feedback is a crucial but time-intensive aspect of the design process. In practice, achieving a mature and convincing design often requires multiple iterations, particularly for interactive prototypes [5]. GUI mockups not only facilitate the exploration of alternative design solutions early in the process [6] but also help refine requirements [7]. Consequently, accelerating mockup creation can significantly reduce both development time and costs [8, 9].

The refinement process often entails spending a significant amount of time on programming tasks, such as adjusting HTML, CSS, and ensuring proper alignment of elements to the requirements. These tasks, while essential, can become cumbersome and divert attention from the creative and human-interaction-related aspects of the design process. In this context, leveraging Large Language Models (LLMs) can help offload these time-consuming programming tasks from designers. By automating the generation of code, LLMs enable designers to dedicate more time to the creative and human-intensive elements of the design process. This shift in focus allows for more time spent on iterating on ideas, refining the user experience, and ensuring that the final design aligns with both user needs and stakeholders' objectives.

Recent advancements in deep learning have been leveraged to automate GUI mockup generation. In [8] a deep learning framework has been proposed to convert hand-drawn GUI sketches into Android-based GUI prototypes. Similarly, object detection models have been employed to identify UI elements and their spatial arrangement in high-fidelity UI mockups, generating intermediate representations to automate front-end code generation [9]. In [10] a data-driven GUI prototyping approach was employed to retrieve reusable mobile app GUIs from a large-scale repository using Natural Language Processing (NLP) interfaces. LLMs have also been explored for GUI design. A recent study [11] investigates the feasibility of using LLMs to generate GUI prototypes by producing HTML and CSS code from high-level textual GUI descriptions. However, a recent survey from Stige *et al.* [12] stresses that there are limited contributions focusing on user-centred approaches, and "there is still limited work on how AI can be introduced into processes that depend heavily on human creativity and input". Unlike previous work, our study presents a practical experience of applying these AI-driven approaches in an industrial case study, where mockups were generated in a real-world scenario. Furthermore, while existing approaches primarily focus on using AI for code generation, our approach also leverages LLMs for requirement analysis, enabling the automatic transformation of textual requirements into structured NL-based interfaces.

In this paper, we report our experience in iteratively using LLMs to generate GUI mockups within a user-centred design process involving Trenord[1], a railway operator in Northern Italy. Our approach began with requirement elicitation through focus groups with stakeholders, analysis of existing documentation, and the manual creation of preliminary mockups. During this phase, requirements were analysed and documented to establish a clear foundation for advanced mockup development. We then structured our approach into three key tasks: (i) extracting a list of sections and defining the information architecture based on the documented requirements, (ii) generating HTML and CSS code to create navigable mockups for each section, and (iii) refining the generated code to ensure that the GUI aligned with both the documented requirements and specific stakeholder requests. Throughout each task, the outputs were evaluated by human experts, ensuring a robust human-in-the-loop process. This iterative evaluation and refinement process allowed us to continuously improve the mockups, ensuring they met the stakeholders' needs and expectations. This process was applied to the design of a dashboard supporting predictive maintenance in railways, intended for integration into an existing railway diagnostic portal used by Trenord. Based on our experience, we discuss key lessons learned

[1] https://www.trenord.it/

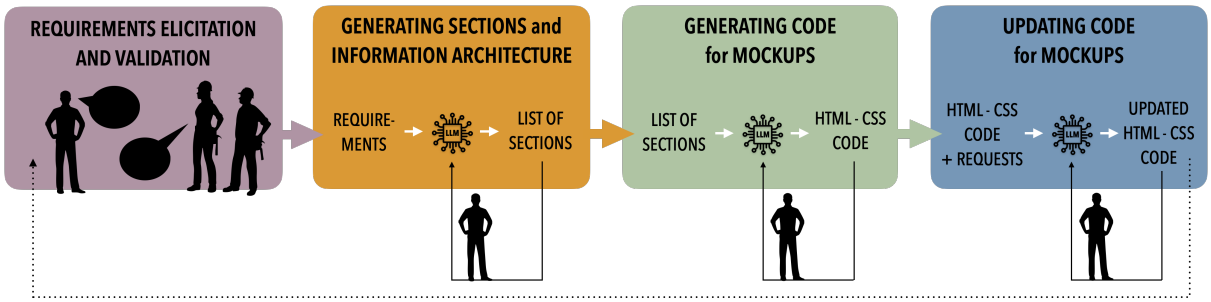


Figure 1: Diagram illustrating the different phases of the proposed approach.

from this approach.

The remainder of this paper is structured as follows. Section 2 introduces the context of the case study. Section 3 details the process used and its application to our case study. Section 4 presents key lessons learned from our experience. Finally, Section 5 discusses the threats to validity and Section 6 concludes the paper and outlines directions for future work.

Online Resources. The generated mockups, the prompts used, and the requirements are available in [13].

2. Context of the Experience

Company and Objectives. Trenord is a railway transport company operating in Northern Italy, managing a fleet of over 400 trains. Each Trenord train is equipped with a number of onboard devices, each generating diagnostic data. An onboard diagnostic platform collects and stores this data, transmitting it to a wayside system for further analysis. Trenord personnel can access diagnostic data, fleet status, alarm history, and statistical reports through a web-based diagnostic portal designed for helpdesk and engineering teams. In our collaboration with Trenord, we are working on predictive maintenance techniques that can use diagnostic data to predict future failures and anticipate maintenance needs[14]. During the collaboration, the need to develop a graphical dashboard was highlighted. This dashboard would provide visualised predictive data and detailed insights on the current train logs, covering both the entire fleet and individual trains. It will display the results of two different prediction methodologies. The dashboard is intended for two classes of users: maintenance personnel and engineering personnel. It will be integrated into the web-based diagnostic portal as a separate and dedicated component.

The dashboard design process involves three stakeholders from Trenord's engineering team, and two researchers. The stakeholders are the final users, and are thus responsible for providing domain knowledge and requirements, overseeing mockup creation, and ensuring that the final product meets their needs. The two researchers are responsible for the requirements elicitation, mockups creation, and refinement. The collaboration experience reported in this paper is composed of two main phases: a *preliminary phase*, in which requirements are elicited through traditional techniques, i.e., document analysis, think-aloud, and manual GUI prototyping; a *consolidation phase*, in which mockups are automatically generated using LLMs. In the following, we briefly describe the preliminary phase. In Sect. 3, we illustrate the consolidation phase, which is the focus of this work. A visual representation of the approach is provided in Figure 1.

Preliminary Phase. To facilitate ongoing collaboration and ensure continuous alignment with the stakeholders' needs, the team—composed of the stakeholders and the designers—met every two weeks in recorded online meetings. In this phase, 6 meetings have been carried out. Initially, the stakeholders provided the designers with the user manual for the existing diagnostic portal to help inform the design and development of the new additional dashboard.

To initiate the design process, the designers conducted a 1-hour focus group with the stakeholders, where they interacted with the existing portal and discussed their regular tasks. This *think-aloud*

session provided valuable insights into the stakeholders' workflow, how they interact with the current diagnostic portal, and the main functionalities they are interested in, which guided the early stages of dashboard design.

The second step in the process involved the *manual* creation of initial mockups by the designers. The mockups were developed based on data gathered from existing predictive maintenance methodologies applied to the Trenord fleet, as well as insights into stakeholders' tasks and workflows. These preliminary designs provided a visual representation of the predictive maintenance dashboard, serving as a foundation for further refinement and iteration. Upon review, the stakeholders provided valuable feedback, highlighting key functionalities they desired, as well as additional features not initially included in the mockups.

In parallel, a requirements document was drafted based on the feedback and insights gathered during the mockup review phase. The document, outlining the functional and non-functional requirements for the dashboard, was revised by the stakeholders. This iterative process of revision helped to clarify the requirements and ensure that all aspects of the dashboard design were well-defined and addressed the stakeholders' goals. The drafted document served as the foundation for the next phase of mockup generation.

3. Mockup Generation Experience

In this phase, we investigated how LLMs can support the GUI design process by automating specific tasks. In particular, we identified and explored three tasks where LLMs can provide assistance: (1) requirements analysis and information organisation, (2) mockup generation, and (3) mockup refinement.

Each task involves providing an input to the LLM (in the form of a prompt), which then generates an output to support the designer.

We performed the tasks sequentially, using the output of each task as the input for the next, while allowing for iterative refinements. This approach enabled a flexible and adaptive design process, ensuring that each stage informed and improved the subsequent steps.

We tested both ChatGPT[2] and DeepSeek[3]. The free version of ChatGPT provided limited daily access to GPT-4o, requiring a new chat session once the limit was reached, disrupting iteration and losing context. It then defaulted to GPT-3.5, which proved less effective for generating structured and high-quality HTML and CSS code. To overcome these limitations, we also experimented with DeepSeek-V3, which does not impose such restrictions, allowing for a more continuous and iterative workflow. To achieve better results, we followed OpenAI's guidelines for prompt engineering[4]. As part of this approach, we instructed the model to adopt a specific persona to enhance the relevance and quality of the generated outputs. Specifically, all prompts used in our study began with the directive: "Please act as a GUI designer". This guided the LLM to generate responses that aligned with best practices in user interface design, facilitating the creation of well-structured and user-centred mockups.

Below is a description of each task based on our experience throughout the process, accompanied by excerpts of the prompts used and the output generated by the LLM. The full set of prompts for all three tasks, along with the corresponding output, are provided in [13].

Task 1 Generating Sections and Information Architecture. The first task focuses on transforming requirements into a structured interface architecture. For this task, we leveraged ChatGPT to generate an initial list of sections for the dashboard, ensuring that each section encapsulates relevant data and features while maintaining logical navigation paths. We operated iteratively in this phase, refining and adjusting the generated sections as needed. Below is the input prompt used in the context of this study (for brevity, the full requirements are provided in [13]).

[2]https://chatgpt.com/

[3]https://chat.deepseek.com/

[4]https://platform.openai.com/docs/guides/prompt-engineering

Please act as a GUI designer. Below are the requirements for the design of a dashboard for predictive maintenance in railways. The dashboard shall be included into a web-based diagnostic portal. The users shall be able to access the dashboard through a dedicated button in the Home page of the portal.
Please, provide a list of sections for the dashboard based on these requirements. Each section should include related data and features and be accessible from at least one other section. Please, provide the list with a short explanation of the content each section should include and an indication of how to reach it from the other sections.
Requirements:
1. Introduction
This document specifies the requirements for a Predictive Maintenance Dashboard designed to support maintenance operations and engineering analysis across the TSR train fleet.
2. Non-Functional Requirements
2.1 Accessibility and Compatibility
NFR-001: The dashboard shall be accessible via modern web browsers (Chrome, Firefox, Edge, Safari).
(...)

As output, the LLM provided a list of the dashboard sections, including a description of the content shown in each section and details on how to navigate between them.

The output provided has been iteratively refined, as the sections were not entirely accurate due to some requirements being incomplete or imprecise. This process allowed us to not only improve the section generation but also refine the requirements, ensuring that they more effectively supported the design and implementation of the dashboard. Table 1 presents the list of the sections and their description.

Task 2 Generating Code for Mockups. The second task focuses on generating code to visualise mockups based on the structured list of sections provided by Task 1, each containing relevant content and navigation paths. The LLM is leveraged to generate HTML and CSS code, facilitating the rapid creation of mockups that are both easily visualisable and navigable. For this task, we initially leveraged ChatGPT for the first three sections but then transitioned to DeepSeek due to the free-plan limitations of ChatGPT, as discussed earlier in the paper.

The prompts used for this task follow the same structure as the one shown below, which was used to generate the code for the section "3. Prediction Detail View". For navigation purposes, we provide LLM with the generated code for the section "2. Train Configuration View" (i.e., trainView.html in the prompt below). This allows the LLM to establish the appropriate navigation paths between the two files, ensuring a structured and interconnected navigation flow across sections.

Please act as a GUI designer. Please, generate the HTML and CSS code for the page described below (3. Prediction Detail View) in a single file. This page is intended for a predictive maintenance dashboard for railways and displays the details of a predicted error. It is accessible when an element <tr> with the class status-red or status-yellow from the table in the attached file (trainView.html) is clicked.
3. Prediction Detail View
Content:

- Error Details (Type, description, affected component)
- Affected Train Details (train ID, carriage ID)
- Prediction Time

Navigation:

- Accessible from Train Configuration View by selecting a component
- Links to Maintenance Personnel View and Engineering View for further actions

As an output, the LLM provided an HTML and CSS file for section "3. Prediction Detail View". To better align with the existing sections and meet the designers' expectations, such an initial version was refined through further iterations in Task 3. This process highlights the essential role of human creativity in shaping the final design, ensuring coherence and usability beyond the LLM's initial output. The complete set of generated sections, including both the initial versions from Task 2 and the revised versions from Task 3, is available in [13].

Task 3 Updating Code. The third task focuses on refining existing HTML and CSS code. Refining mockups is a crucial step in GUI design, particularly in our case, where the mockups are generated,

as it ensures alignment with design expectations, stakeholder needs, and overall system coherence. Since LLMs facilitate rapid iteration cycles by quickly modifying existing HTML and CSS code based on specified requests, creativity in the refinement process plays a more prominent role, allowing designers to experiment with variations, explore new ideas, and fine-tune details without being constrained by technical implementation. Furthermore, LLMs can assist in implementing technical elements but often require human input to introduce creative solutions and ensure that the design truly reflects the stakeholders' vision.

Below are two of the prompts used to update the code for the section "3. Prediction Detail View".

> Please update the code so that the layout of the page is as the one in trainView.html. The title of the page shall be positioned on top of the page on the grey background and the section with the train information shall be positioned exactly as in trainView.hmtl.

> Please add the following svg icon, filled in white, to the engineering view button
> `<svg code for the icon>`
> and the following svg icon, filled in white, to the maintenance view button
> `<svg code for the icon>`

As output, the LLM provided the updated HTML and CSS code incorporating the requested modifications (see [13] for the complete set of generated mockups).

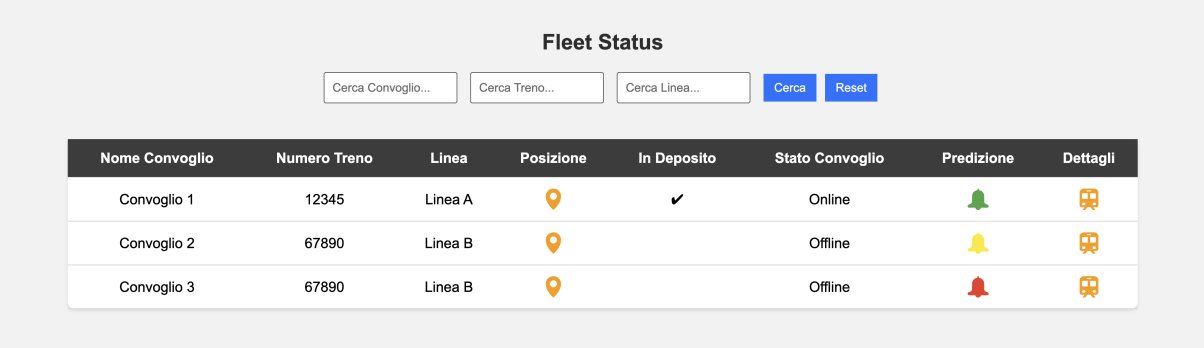
In total, six mockups were generated. Figure 2 displays two of these, the section that shows the fleet status and the section that shows the detail of a single train. Table 1 provides further details on each section mockup, including its description, the number of iterations required to finalise it, and the LLM used in the process.

Generated Mockup	Description	Iterations	LLM used
1. Fleet Overview	Section showing the status of the entire train fleet. For each train shows the prediction indicator	4+5	ChatGPT + DeepSeek
2. Train Configuration View	Section displaying the status of a single train, providing detailed information on predicted errors for each component.	8+7	ChatGPT + DeepSeek
3. Prediction Detail View	Section presenting the detail of the predicted error.	1 + 3	ChatGPT + DeepSeek
4. Maintenance Personnel View	Section that enables the maintenance personnel to manage the predicted error.	3	DeepSeek
5. Engineering Personnel View (prediction done with ML)	Section that enables the engineering personnel to check the details of the predicted error.	3	DeepSeek
6. Engineering Personnel View (prediction done with Fault Tree Analysis)	Section that enables the engineering personnel to check the details of the predicted error.	1	DeepSeek

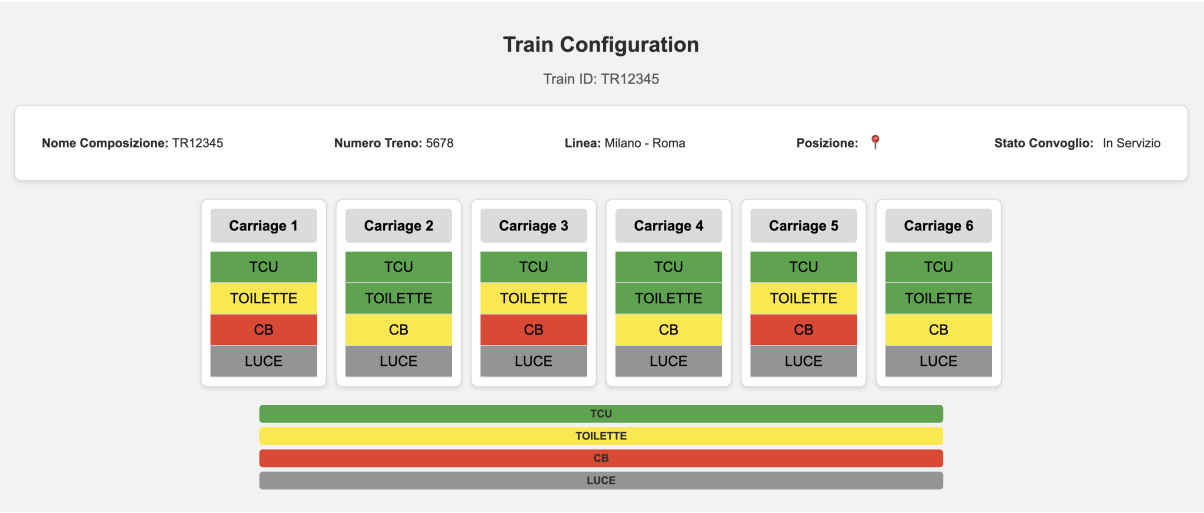
Table 1
Details of the generated mockups, including descriptions, the number of iterations performed with the LLM to produce the final mockup, and the LLMs used. When both LLMs were utilised, the iteration counts indicate the number of iterations conducted with ChatGPT followed by those with DeepSeek.

4. Feedback from Stakeholders and Lessons Learned

After applying the procedure illustrated in Section 3, we generated six mockups representing six different sections of the dashboard (see Table 1). These mockups were presented and discussed in a focus group with the stakeholders. During the meeting, the mockups were shown to the stakeholders as an operational dashboard, demonstrating the workflow for two different classes of users (maintenance personnel and engineering personnel). We then asked the stakeholders to provide feedback on each of the generated sections. Additionally, they were prompted to reflect on the following questions: *How do these interfaces compare to the ones you initially saw? How do you find this approach compared to the previous (manual) one in terms of satisfaction with the result? What advantages and benefits do you see with this procedure? and what disadvantages? Are these interfaces informative enough for you to indicate*



(a) Section showing the status of the entire train fleet.



(b) Section showing the status of a single train.

Figure 2: Two generated mockups composing the predictive maintenance dashboard.

changes? What is your perspective on the approach in terms of creativity? Do the generated mockups inspire you and spark new ideas, or do they limit your creativity by presenting solutions that feel already established?

Regarding the comparison between the manual and generated interfaces, one of the stakeholders remarked: *The new interfaces give the impression of something more concrete, with slightly more refined graphics, more in line with our portal.* All three stakeholders agreed with this assessment. They found the new interfaces to be more polished and structured, appreciating the enhanced clarity and realism of the mockups. They noted that the new interfaces brought the system closer to their expectations.

When asked about the approach used to generate the interfaces, one of the stakeholders initially commented *I am scared that DeepSeek was able to do this.* However, they ultimately found this approach to be more concrete and practical than the manual one. They described it as *certainly more concrete and practical; you immediately see the graphic result of what you asked for in the requirements. You can quickly identify its potential, highlight areas for improvement, and it also supports the development of the requirements themselves.*

Regarding the advantages of the approach, the stakeholders emphasised that the iterative process enabled more immediate and comprehensive requirement definitions. One stakeholder remarked, *A more complete requirement comes first, without having to wait for the supplier to implement a first version. I can request changes immediately, making the process more practical, faster, and easier to manage.* All stakeholders agreed that this approach would simplify both the technical and administrative aspects, leading to more efficient contract management. Regarding potential disadvantages, the stakeholders cautioned that this approach might lead to unrealistic expectations, as users could overestimate the

actual capabilities of the system based solely on its convincing graphical representation. One stakeholder remarked, *You have to keep your feet on the ground for a moment.* Additionally, the same stakeholder recognised that the main disadvantage of this approach would likely impact suppliers more than stakeholders, as the ability to generate mockups independently could reduce the suppliers' role in the process.

When asked about the clarity of the interfaces, the stakeholders confirmed that the mockups were informative enough to indicate necessary changes. They were able to provide several modifications and feedback during the session, and when asked *Are these interfaces informative enough for you to indicate changes?*, one of the stakeholders said, half-smiling: *Well, it seems to me that we have discussed enough changes.* This demonstrated the effectiveness of the generated mockups in supporting rapid iteration and refinement.

For what concerns creativity, the discussion with the stakeholders revealed that, while one initially might assume that using LLMs for mockup generation could limit creativity by presenting already established solutions, in most cases, this approach actually fosters creativity. According to one stakeholder, *The real creative process happens when we define the requirements that we feed into the LLM. That's when we have to conceptualise the functionalities and visual aspects of the portal. The LLM then takes care to actually realising what we just imagined.* Rather than restricting creativity, LLMs help bring abstract ideas to life, allowing for continuous iteration and refinement. *Seeing a concrete realisation of our ideas enables us to revisit our requirements, modify them, and integrate missing elements.* Ultimately, they all agree that the iterative nature of the approach ensures that rather than being constrained by predefined solutions, designers are empowered to explore and refine their ideas dynamically.

Beyond these considerations, insights from the focus group discussion, combined with our observations during the iterative process, highlighted a number of lessons learned, which are summarised below.

- **Requirements Quality Matters:** The quality of the requirements in terms of clarity and absence of ambiguity helps to generate well-designed interfaces. Without this clarity, multiple iterations may be needed to reach the desired level of quality in the output. For instance, uncertainties regarding how to present and compare the results of two different prediction methods led to several iterations in generating the section that displays the details of the train, where both predictions are presented. This phenomenon—which can be summarised as "garbage in-garbage out"—was also observed in recent studies, in which requirements were used as sources to generate UML models [15], and traceability links [16].
- **Human Creativity Matters:** One of the key advantages of using LLMs in the design process is their ability to offload time-consuming, clerical tasks such as programming and structuring the content, allowing designers to dedicate more time to the creative and human-interaction-related aspects of the design process. By automating the generation of code, LLMs enable designers to quickly explore different stylistic variants of the same mockup, testing multiple creative solutions for a given section without the need to manually code each variation. This allows designers to experiment with diverse approaches, providing the freedom to creatively iterate without getting bogged down in technical details. Furthermore, by streamlining coding activities, designers can spend more time engaging with stakeholders to elicit requirements, gather feedback, and refine mockups iteratively. These interactions, which are critical for ensuring that the final design aligns with both user needs and stakeholders' objectives, risk being under-emphasised when designers are overwhelmed with technical tasks.
- **LLM Choice Matters:** The choice of model significantly impacts the results, as both ChatGPT and DeepSeek have their advantages and limitations.
  - *Content Accuracy:* ChatGPT tends to modify the requested content across iterations, often introducing unintended changes that can require additional effort to correct. This issue becomes particularly time-consuming when multiple iterations are needed, as each round may introduce slight deviations from the original specifications, forcing the designer to repeatedly refine and realign the output. In contrast, DeepSeek-generated files strictly

adhere to the specified content from the first attempt, minimising the need for iterative corrections. This consistency significantly reduces the overall time spent on revisions, making DeepSeek a more efficient choice when content accuracy is critical.

- *Design Quality:* ChatGPT tends to generate basic and often simplistic designs (e.g., white background with black text), resulting in schematic and less visually appealing layouts that require additional iterations to refine. In contrast, DeepSeek produces more visually polished designs from the outset, incorporating modern frameworks and reducing the number of refinement cycles needed. In our experience, DeepSeek proved to be more suitable for our purposes, as the dashboard is intended for railway operations, where a modern yet functional design is preferred without an excessive emphasis on creativity. This is especially important since the dashboard must seamlessly integrate into an existing portal. However, in cases where the designer wishes to have full creative control over the design, relying on DeepSeek's pre-generated styles could be limiting.
- *Workflow Constraints:* Using ChatGPT on a free plan means relying on GPT-4o, the most powerful version, but with daily usage limits. Once these are exceeded, the fallback to GPT-3.5 results in lower-quality outputs, disrupting the workflow. DeepSeek, while free and performant, frequently struggles with server overload, preventing tasks from being completed reliably.
- Live Coding Functionality: The latest version of ChatGPT displays the generated code in a separate canvas next to the request, allowing for a live preview of the output. This feature enables fast iterations, which can enhance the workflow with stakeholders. In contrast, DeepSeek lacks this functionality, requiring designers to save the file locally and open it in a browser to view the results.

Ultimately, the choice between GPT and DeepSeek depends on the specific needs of the project–whether prioritising content accuracy, design quality, workflow stability, or live coding functionality.

- **Prompting Style Matters:** When executing Tasks 2 and 3 (code generation and refinement), maintaining sequential prompts within the same chat session is critical, as the LLM retains conversational memory. However, this approach can introduce challenges during multiple iterations. Scattered or disorganised dialogue—where the LLM must reason over fragmented information—often leads to inconsistencies or overlooked details. To mitigate this, prompts should be clearly defined and structured sequentially, with each request isolated for clarity. For example, after generating initial sections, designers can accelerate subsequent tasks by providing existing code as input and instructing the LLM to match its style. This consistency reduces iterations, as the system aligns outputs with the expected format from the start. In our context, we found it beneficial to perform multiple iterations in Task 3, progressively adding details to refine the output while maintaining a coherent workflow. Notably, once the LLM "learned" the coding style of earlier sections, subsequent mockups required fewer adjustments, demonstrating how structured prompting and style consistency synergise to enhance efficiency.

5. Threats to Validity

This is an experience report, so its empirical rigour is inherently limited, as the focus is mainly on lessons learned from practice, and on triggering further research. A more rigorous case study or controlled experiment, based on the outcomes of the current experience, will provide more empirically sound evidence. In the following, we highlight the main threats to validity.

Construct Validity. This experience report aims to identify initial lessons learned from a preliminary exploration, and it is not specifically focused on well-defined constructs to be empirically evaluated. While this limitation cannot be avoided, our findings and reflections provide hints on possible relevant constructs to be systematically assessed in the future. In particular, a main construct that will be considered is user acceptance, which will be evaluated with standard models, e.g., the Technology

Acceptance Model (TAM) [17], as we did in a previous work [18]. Other constructs worth considering are the speed of the generation process, e.g., how many iterations are required to reach a satisfying output, the user satisfaction with the produced mockups, and the *creativity stimulation*, measured, e.g., with the Consensual Assessment Technique (CAT) [19], or the Torrance Tests of Creative Thinking (TTCT) [20].

Internal validity. Our outcomes may be affected by the Hawthorne effect, due to the involvement of the main investigator in the showcase of the results. However, this risk is mitigated by the fact that the considered technology was not developed by the investigator, but is a third-party product.

External validity. Generalisability is limited by the inherent limitations of case studies and experience reports, as highlighted by Stol and Fitzgerald [21]. Furthermore, findings are based on two specific tools, and different results may be obtained with alternative LLMs.

6. Conclusion and Future Work

In this study, we explored the use of LLMs to support GUI design by structuring textual requirements, generating mockups, and refining designs through iterative feedback. Several insights and lessons were derived from our experience.

Our findings suggest that LLMs can enhance the design process by accelerating iterations and providing a concrete visual foundation for discussion. However, they do not replace human creativity; rather, they act as powerful assistants, reducing the time spent on repetitive tasks and enabling designers to focus on refining and personalising the interface. With LLMs handling the programming workload, designers can focus more on strategic decisions, such as refining the user experience and incorporating human-centred design principles, which can lead to more innovative and user-tailored interfaces. This shift empowers designers to prioritise creativity, collaboration, and continuous refinement—key elements that elevate the quality of the final product.

Additionally, LLM-powered GUI generation helps stakeholders gain a clear understanding of the results at a very early stage, streamlining contract management with suppliers.

While we performed these tasks sequentially, they can also be carried out independently, allowing designers to tailor the approach to their specific needs—whether extracting structured insights, creating initial mockups, or refining layouts based on stakeholder input.

Future work could explore the generation of multiple GUI variants based on different layout styles to further assess the adaptability of LLMs in diverse design contexts. Additionally, the generated mockups could be leveraged to automatically refine and enhance the requirements, creating a more seamless and iterative design workflow.

References

[1] J. D. Gould, C. Lewis, Designing for usability: key principles and what designers think, Communications of the ACM 28 (1985) 300–311.

[2] A. Davis, O. Dieste, A. Hickey, N. Juristo, A. M. Moreno, Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review, in: 14th IEEE International Requirements Engineering Conference (RE'06), IEEE, 2006, pp. 179–188.

[3] M. Bano, D. Zowghi, A systematic review on the relationship between user involvement and system success, Information and software technology 58 (2015) 148–169.

[4] D. Stone, C. Jarrett, M. Woodroffe, S. Minocha, User interface design and evaluation, Elsevier, 2005.

[5] T. R. Silva, J.-L. Hak, M. Winckler, O. Nicolas, A comparative study of milestones for featuring GUI prototyping tools, Journal of Software Engineering and Applications 10 (2017) 564–589.

[6] D. Baumer, W. Bischofberger, H. Lichter, H. Zullighoven, User interface prototyping-concepts, tools, and experience, in: Proceedings of IEEE 18th International Conference on Software Engineering, IEEE, 1996, pp. 532–541.

[7] M. Brhel, H. Meth, A. Maedche, K. Werder, Exploring principles of user-centered agile software development: A literature review, Information and software technology 61 (2015) 163–181.
[8] A. A. Abdelhamid, S. R. Alotaibi, A. Mousa, Deep learning-based prototyping of Android GUI from hand-drawn mockups, IET Software 14 (2020) 816–824.
[9] M. Samir, A. Elsayed, M. I. Marie, A model for automatic code generation from high fidelity graphical user interface mockups using deep learning techniques., International Journal of Advanced Computer Science & Applications 15 (2024).
[10] K. Kolthoff, C. Bartelt, S. P. Ponzetto, Data-driven prototyping via natural-language-based gui retrieval, Automated software engineering 30 (2023) 13.
[11] L. Fiebig, K. Kolthoff, C. Bartelt, S. P. Ponzetto, Effective GUI generation: Leveraging large language models for automated GUI prototyping, in: Proceedings of the 58th Hawaii International Conference on System Sciences, 2025.
[12] Å. Stige, E. D. Zamani, P. Mikalef, Y. Zhu, Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda, Information Technology & People 37 (2024) 2324–2352.
[13] G. Broccia, A. Borselli, M. R. Cefaloni, F. Delcorno, A. Ferrari, An Experience Report on Leveraging LLMs for GUI Generation: Automating Coding to Prioritise Creativity - Replication Package, 2025. URL: https://doi.org/10.5281/zenodo.14871563.
[14] R. Ferdous, G. Spagnolo, A. Borselli, L. Rota, A. Ferrari, Identifying maintenance needs with machine learning: a case study in railways, in: 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW), IEEE, 2024, pp. 22–25.
[15] A. Ferrari, S. Abualhaijal, C. Arora, Model generation with LLMs: From requirements to UML sequence diagrams, in: 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW), IEEE, 2024, pp. 291–300.
[16] A. Vogelsang, A. Korn, G. Broccia, A. Ferrari, J. Fischbach, C. Arora, On the impact of requirements smells in prompts: The case of automated traceability, arXiv preprint arXiv:2501.04810 (2025).
[17] N. Marangunić, A. Granić, Technology acceptance model: a literature review from 1986 to 2013, Universal access in the information society 14 (2015) 81–95.
[18] G. Broccia, M. H. ter Beek, A. L. Lafuente, P. Spoletini, A. Fantechi, A. Ferrari, Evaluating the understandability and user acceptance of attack-defense trees: Original experiment and replication, Information and Software Technology 178 (2025) 107624.
[19] T. M. Amabile, Social psychology of creativity: A consensual assessment technique., Journal of personality and social psychology 43 (1982) 997.
[20] E. P. Torrance, Torrance tests of creative thinking, Educational and psychological measurement (1966).
[21] K.-J. Stol, B. Fitzgerald, The abc of software engineering research, ACM Transactions on Software Engineering and Methodology (TOSEM) 27 (2018) 1–51.