Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10288

ReqGPT: a fine-tuned large language model for generating requirements documents

Kata Amanda Schiller, Meno-Said Haddad and Arthur Seibel,✉

Leuphana University Lüneburg, Germany

✉ arthur.seibel@leuphana.de

ABSTRACT: Effective product development relies on creating a requirements document that defines the product's technical specifications, yet traditional methods are labor-intensive and depend heavily on expert input. Large language models (LLMs) offer the potential for automation but struggle with limitations in prompt engineering and contextual sensitivity. To overcome these challenges, we developed ReqGPT, a domain-specific LLM fine-tuned on Mistral-7B-Instruct-v0.2 using 107 curated requirements lists. ReqGPT employs a standardized prompt to generate high-quality documents and demonstrated superior performance over GPT-4 and Mistral in multiple criteria based on ISO 29148. Our results underscore ReqGPT's efficiency, accuracy, cost-effectiveness, and alignment with industry standards, making it an ideal choice for localized use and safeguarding data privacy in technical product development.

KEYWORDS: new product development, requirements, machine learning, large language models, fine-tuning

1. Introduction

Technical product development generally begins by clarifying the problem or task, often culminating in a comprehensive requirements list (Pahl et al., 2007; VDI 2221-1, 2019; Schlattmann & Seibel, 2021). This document, often presented in tabular format, details all the essential specifications, functions, and constraints necessary for product development. It is continuously updated throughout the development process and serves as a foundation for design, evaluation, and decision-making.

Compiling the relevant information to create a requirements list is a complex and time-consuming task, involving substantial manual effort (Dąbrowski et al., 2020), particularly for new product development. This process is usually carried out by experts using methods such as market analysis (Palomares et al., 2021), stakeholder surveys (Franch, 2021), product reviews (Lim et al., 2021), user feedback (Van Vliet et al., 2020), and/or quality function deployment (Pokorni et al., 2022). In this context, large language models (LLMs) have recently emerged as a promising tool to support and accelerate the requirements engineering process (Ronanki et al., 2024) as they can understand and process natural language and are able to effectively extract, structure, and refine information from large datasets.

Interacting with LLMs typically involves a process known as prompt engineering (Sahoo et al., 2024), where users craft specific requests to elicit desired responses. However, producing high-quality outputs through this method can be challenging, as it often requires providing extensive context for the request and additional details about the desired result. Furthermore, LLMs are generally sensitive to the phrasing of prompts, meaning that even slight variations in wording can result in significantly different outcomes (Jiang et al., 2020).

To address these challenges and enhance the utility of LLMs for generating requirements lists, we have developed a domain-specific GPT (generative pre-trained transformer) that enables product developers to create high-quality requirements lists for technical products using a single standardized prompt, such as "Create a requirements document for [product]." This document can be tailored to specific contexts and

company needs by incorporating elements such as stakeholder surveys, ISO standards, international and regional regulations, or internal requirements from previous projects (Dehn et al., 2023).

2. Background

2.1. General LLMs

In the field of artificial intelligence, LLMs have become groundbreaking tools, revolutionizing human-machine interactions. These models are distinguished by the massive datasets on which they are trained, enabling them to process and respond to complex requests. Their extensive training allows not only for the comprehension of natural language but also for the interpretation of linguistic relationships and the generation of coherent, contextually appropriate responses.

In contrast to traditional natural language processing (NLP) models, which heavily depend on recurrent or convolutional neural networks (Kombrink et al., 2011), modern transformer-based models—such as GPT-4 from OpenAI—are built entirely on attention mechanisms, particularly self-attention (Vaswani et al., 2017). This architecture facilitates parallel data processing, enabling these models to efficiently extract meaningful information from vast amounts of text. However, while general LLMs are powerful text creators, they may lack the precision and domain-specific knowledge required for specialized tasks.

2.2. Domain-specific LLMs

Domain-specific LLMs are tailored to support specialized tasks across various disciplines. In contrast to general-purpose LLMs, which are designed to handle a wide variety of tasks, domain-specific LLMs are explicitly trained to address the unique requirements and nuances of specific industrial segments. In these specialized contexts, they can even surpass much larger general LLMs for previously unseen tasks (Chung et al., 2024).

Examples of domain-specific LLMs already exist in fields such as finance (Wu et al., 2023; Yang et al., 2023), law (Cui et al., 2023), natural sciences (Xie et al., 2023), biomedicine (Lee et al., 2020), and bio-inspired design (Chen et al., 2024). The models are typically derived from general-purpose LLMs via a process known as fine-tuning (Moradi et al., 2024). Fine-tuning involves retraining or adapting a model using training datasets to improve its performance on targeted tasks.

Several methods exist for fine-tuning LLMs for specific tasks, with full-parameter fine-tuning being the most straightforward approach (Sun et al., 2023). The method involves adjusting all layers of the model by training it on task-specific data, making it particularly effective for scenarios with large and distinct datasets that differ significantly from the original pre-training data. While it allows the model to deeply learn and adapt to new requirements, it has notable drawbacks, including substantial memory demands and the need for high-performance hardware (Ding et al., 2023).

To reduce the computational and memory effort for fine-tuning, low-rank adaptation (LoRA) provides a more efficient alternative (Hu et al., 2021). LoRA operates on the assumption that the weight changes during model adaptation have a low "intrinsic rank." Instead of retraining the entire model, the method optimizes low-rank decomposition matrices that encapsulate these changes in the dense layers. Building on this concept, quantized low-rank adaptation (QLoRA) further minimizes the memory requirements by introducing advanced memory management techniques (Dettmers et al., 2024). QLoRA achieves this by applying four-bit quantization to the LLM and enabling backpropagation of gradients through these quantized, frozen layers, significantly improving efficiency without sacrificing performance.

2.3. LLMs for requirements lists

The creation of a requirements list is a text-based activity, making it highly suitable for the application of NLP models. LLMs can assist in extracting relevant information from extensive pre-trained datasets while addressing ambiguities and inconsistencies. This ensures that requirements are clear, structured, and well-aligned with product development needs by converting project-specific language into formal notations (Bertram et al., 2022) and machine-readable formats (Ray et al., 2023).

While prompt engineering serves as an alternative to fine-tuning for requirements elicitation, it is highly dependent on precise wording. Research in LLM-based prompt engineering recommends incorporating key elements such as intent, context, motivation, structure, output indicators, example implementations, and additional consequences (Giray, 2023; White et al., 2024). Techniques like few-shot and chain-of-

thought prompting methods are particularly useful (Ronanki et al., 2024). However, prompt engineering lacks domain specificity and often yields suboptimal output performance (Gu et al., 2023).
Recent advances propose various approaches to address engineering requirements tasks. For example, LLMs' inherent capability to detect semantic similarities between words and sentences can be applied to summarize contextually identical requirements (Norheim et al., 2024). Multi-layered frameworks can help address issues arising from conflicting or redundant requirements (Malik et al., 2023). By analysing text data from interviews, online platforms, and other databases, LLMs can optimize formatting, length, memory, and wording more effectively compared to humans (Singhal et al., 2023; Talebirad and Nadiri, 2023; Kapoor et al., 2024).
The MARE framework investigates requirements engineering in software development by using a multi-agent system based on LLMs. This system collaborates to perform the entire requirements engineering process, improving the accuracy and quality of requirements specifications (Jin et al., 2024).
To match user needs with their expectations, the Elicitron framework introduces a possibility to discover linguistic intricacies based on the analysis of interviews with LLMs (Ataei et al., 2025). By categorizing needs as either direct or latent, this method aims to identify subliminal expectations, thereby facilitating information extraction from text—a crucial aspect of requirements engineering.

3. Method

3.1. ReqGPT workflow

Figure 1 illustrates the development process of ReqGPT. First, 120 requirements lists for products from the consumer goods sector were automatically created employing few-shot prompting with GPT-4 using a requirements list structure extracted from Hubka et al. (1988). Precisely 107 of these lists satisfied all necessary criteria according to ISO 29148 (2018), providing a robust dataset for the subsequent training. The training was performed on the base model of Mistral-7B-Instruct-v0.2 over nine epochs, ensuring comprehensive data integration and processing efficiency. The training process resulted in a parameter-efficient and domain-specific API designed for generating high-quality requirements lists for technical product development.

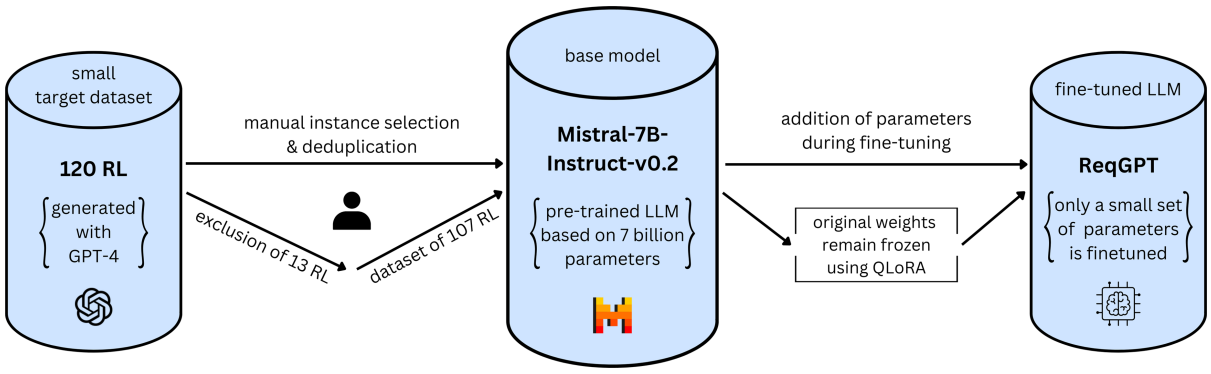

Figure 1. Flowchart of the development process of ReqGPT

The goal of this project was to develop a fine-tuned LLM to enhance the creation of requirements lists for technical product development. The model was not specifically designed to eliminate hallucinations but rather to generate typical requirements for developmental purposes. By connecting the model with external databases, such as standards and regulatory resources, through retrieval-augmented generation (RAG), we expect that hallucinations can be reduced to an acceptable level (Su et al., 2024).

3.2. Data generation

To generate training data for ReqGPT, we synthesized a generalized structure for a requirements list by abstracting from various specific examples within the field of mechanical design (Hubka et al., 1988). The identified requirement types were systematically grouped into categories and further organized into appropriate subcategories. Prompt engineering techniques using GPT-4 were then employed to generate generalized requirements lists. To address the token context window limit of GPT-4 (as of now 8,192), the prompt was divided into two sections. The entire composite prompt is provided in Appendix A.

To optimize the prompt for generating consistent requirements lists, we performed several iterations of refinement, including manual instance selection (Blachnik et al., 2020) as well as deduplication (Lee et al., 2021). During this process, some subcategories were deemed inadequate as they produced irrelevant, misleading, or inconsistent outputs. These outputs were manually refined until the desired results were achieved. This refinement ensured that the input clearly defined individual requirements, avoiding the generation of instruction manuals within the output.
For the dataset, we compiled a sample of 120 consumer products to generate requirements documents. Of these, 13 products deviated strongly from the rest and were therefore excluded from further analysis. The remaining 107 documents were curated and manually refined to ensure consistency and relevance.

3.3. Model selection

The selection criteria for the base model used for fine-tuning focused on open-source accessibility and the ability to perform both training and inference on consumer-grade hardware. At the time of selection, the best-performing model that fulfilled these criteria was Mistral-7B-Instruct v0.2. With just over seven billion parameters, this model works seamlessly with next-generation lightweight notebooks, making it ideal for deployment in typical business environments.
The instruction-tuned variant of Mistral-7B was selected as the base model, as fine-tuning small LLMs with task-specific instructions has been shown to enhance performance on new, unrelated tasks in both zero-shot and few-shot scenarios without increasing computational resources and costs (Ouyang et al., 2022). Given the limited size of the dataset, specific commands were integrated into the model instead of performing full-parameter fine-tuning. This method reduced training time while delivering improved results with fewer input tokens.

3.4. Model training

As outlined in the previous section, Mistral-7B-Instruct-v0.2 serves as the base model for ReqGPT. The fine-tuning process was conducted using an NVIDIA A10 GPU with 24 GB VRAM. The learning rate was set to $3e^{-5}$, and the AdamW optimizer was employed (Kingma & Ba, 2014). Later stages of training incorporated QLoRA with four-bit quantization and an adapter rank of 16.
The fine-tuning process was carried out with 107 curated requirements lists, completed over nine epochs. Training was marked by smooth transitions and incremental improvements, ensuring consistency in the outputs. The model achieved a validation loss value of 0.4332, which was smoothed to 0.4419 after 350 steps. Notably, the threshold for an acceptable validation loss (< 1) was already reached after processing 75 lists, adhering to the characteristics of requirements according to ISO 29148 (2018).
Following training, ReqGPT exhibited a strong coherence and replicability, effectively generating high-quality requirements lists for defined products. Additionally, significant prompt shortening (Patil et al., 2023) augmented the efficiency of the model, enabling the generation of detailed requirements lists with input prompts as concise as 15 tokens. Data for training, analysis, and output are accessible at GitHub.[1]

3.5. Evaluation

To evaluate the output quality of ReqGPT, the corresponding requirements lists were compared to those produced by GPT-4 (the source of the training data) and the baseline Mistral model (prior to fine-tuning). Since language quality is best assessed by human evaluators, we conducted a qualitative study involving 18 graduate-level master students specializing in product design. These participants represented diverse technical disciplines, such as engineering, management, and business informatics.
Participants were tasked with acting as product developers improving a technical product by evaluating multiple features of the requirements lists. The assessment included nine criteria according to ISO 29148 (2018): necessity, appropriateness, correctness, creativity, completeness, coherence, unambiguousness, verifiability, and uniformity of requirements and sub-requirements.
The evaluation was conducted during a 60-minute session where the participants rated the lists on a five-point Likert scale (1: strongly disagree, 2: disagree, 3: neutral, 4: agree, 5: strongly agree). Subsequently, the authors analysed the responses to assess linguistic, semantic, and structural differences between the outputs of the different LLMs. The evaluation results are summarized in Table 1.

[1] https://github.com/IPTS-PRODUCT-DESIGN/ReqGPT

Table 1. Weighted mean scores for requirements lists generated by Mistral, GPT-4, and ReqGPT

Evaluation criteria	Weighted mean score of criteria for each list in %		
	Mistral	GPT-4	ReqGPT
Necessity	22	74	86
Appropriateness	34	70	72
Correctness	29	68	80
Creativity	34	63	70
Completeness	31	67	77
Coherence	42	69	69
Unambiguousness	39	62	74
Verifiability	54	76	82
Uniformity	24	69	79
Average	34	69	77

The study's results show that ReqGPT outperforms both the baseline LLM and GPT-4 in overall mean score performance across the measured criteria. ReqGPT achieved the highest similarity score at 77% with a moderate interquartile range (IQR) of 17% (69–86%), indicating a consistent evaluation across all nine criteria. GPT-4 followed with a similarity score of 69% and a slightly narrower IQR of 13% (62–75%), reflecting a slightly smaller variability in output quality compared to ReqGPT. In contrast, Mistral recorded the lowest similarity score at 34% and exhibited the widest IQR of 32% (22–54%), highlighting significant inconsistencies and, comparatively, the poorest outputs. A representative output from ReqGPT can be found in Appendix B.

Overall, ReqGPT outperformed GPT-4 and Mistral, providing more accurate and consistent results, as reflected by its higher median and lower variability. Furthermore, the p-value for the evaluation criteria across all lists was approximately $< 1\%$, highlighting the statistical significance of ReqGPT's superior performance compared to the other LLMs. The p-value was determined using a two-sided test.

4. Challenges and limitations

4.1. Hallucinations

Current LLMs are generic tools that can generate texts, but this ability alone is insufficient for creating adequate requirements documents. While text processing is a prerequisite for formulating requirements lists, domain specificity is indispensable for a successful application in product development. ReqGPT, however, is not able to mitigate hallucinations in its current form. This requires a detailed understanding of the market situation, the company's internal constraints, legal requirements, as well as relevant norms and standards. To realise this, ReqGPT needs to be extended by relevant databases using RAG methods (Su et al., 2024), which is intended for future research.

4.2. Training dataset

The initial dataset of 120 requirements lists for model retraining was considerably small and was further reduced to 107 lists for few-shot data generation. This reduction was justified by the observation that an acceptable validation loss was reached after training with just 75 lists, rendering the additional 32 lists less necessary. A limited dataset was possible, as it was meticulously curated and deliberately selected upfront, rather than relying on randomized data generation through the LLM. For future replications of similar fine-tuning processes, careful monitoring of the validation loss is advised, as it serves as a key indicator for effective learning while minimizing the risk of overfitting (Schubert et al., 2024).

4.3. Bias in human evaluation

The analysis of LLM outputs through human evaluation inherently involves a degree of subjectivity and can lead to biased results (Tjuatja et al., 2024). To address this limitation, incorporating a more diverse group of evaluators—beyond engineering master students—could provide a broader and more balanced assessment. Expanding the cohort of evaluators would improve the clarity and precision of the analysis, leading to more robust and significant outcomes.

4.4. LLMs for evaluation

An alternative evaluation approach involves using multiple-agent collaborations of LLMs as impartial evaluators. Unlike human evaluation, which is both time-consuming and costly, LLM-based evaluation addresses these challenges more efficiently (Dubois et al. 2024). Research has shown that humans often focus less on output correctness and more on relatively marginal aspects, such as geometry, potentially missing critical issues (Singhal et al., 2023). LLMs, by contrast, provide the possibility for more accurate assessments with fewer errors, enhancing the reliability of evaluation outcomes (Chang et al. 2024).

5. Conclusion and future work

This study demonstrates that ReqGPT, even though it is based on the smaller Mistral-7B-Instruct-v0.2 model, can outperform the significantly larger, general-purpose GPT-4 in generating requirements lists for technical product development. Despite the larger scale and broader data access of GPT-4, ReqGPT's targeted adaptations enable it to better address industry-specific demands.

The findings highlight several important advantages of smaller LLMs. First, they operate with increased efficiency, requiring less computational power and functioning effectively on local hardware. Second, local deployment of LLMs enhances data privacy and cybersecurity, which are essential for industries handling sensitive or proprietary information. By minimizing reliance on external networks and servers, local models reduce data exposure risks while also offering stability in environments with inconsistent internet connectivity (Da Silva et al., 2022). Third, smaller LLMs are easier to customize and specialize for specific tasks or industries. Models like Mistral-7B-Instruct-v0.2 can be fine-tuned with greater ease, enabling improved performance in targeted applications and generating outputs that are highly relevant and customized to the user's operational context.

The results from this study pave the way for two critical avenues for future research. First, efforts should focus on integrating LLM-accelerated workflows into real-world product development processes. This involves refining the models to improve their specificity and accuracy and embedding them in existing product development workflows to evaluate their practical impact. Second, incorporating agent-based retrieval systems offers significant potential to improve LLMs' contextual understanding. These systems would extract specific data from knowledge databases, thus enriching the models' context with relevant, up-to-date information, improving both the quality and relevance of the content produced.

Acknowledgement

We would like to thank the participants of the master course "AI-supported Product Development" from Leuphana University Lüneburg for taking part in the evaluation.

References

Ataei, M., Cheong, H., Grandi, D., Wang, Y., Morris, N., & Tessier, A. (2025). Elicitron: a large language model agent-based simulation framework for design requirements elicitation. *Journal of Computing and Information Science in Engineering*, 25(2), 021012.

Bertram, V., Boß, M., Kusmenko, E., Nachmann, I. H., Rumpe, B., Trotta, D., & Wachtmeister, L. (2022). Neural language models and few-shot learning for systematic requirements processing in MDSE. In *Proceedings of the 15th ACM SIGPLAN International Conference on Software Language Engineering (SLE)* (pp. 260–265). https://doi.org/10.1145/3567512.3567534

Blachnik, M., & Kordos, M. (2020). Comparison of instance selection and construction methods with various classifiers. *Applied Sciences*, 10(11), 3933. https://doi.org/10.3390/app10113933

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., ... & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3), 1-45. https://doi.org/10.1145/3641289

Chen, L., Cai, Z., Jiang, Z., Luo, J., Sun, L., Childs, P., & Zuo, H. (2024). AskNatureNet: a divergent thinking tool based on bio-inspired design knowledge. *Advanced Engineering Informatics*, 62, 102593.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., ... & Wei, J. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70), 1-53. http://jmlr.org/papers/v25/23-0870.html

Cui, J., Li, Z., Yan, Y., Chen, B., & Yuan, L. (2023). ChatLaw: open-source legal large language model with integrated external knowledge bases. https://doi.org/10.48550/arXiv.2306.16092

Da Silva, P. H., Benitti, F., & Wangham, M. (2022). Framework for the development of computational solutions for the support of requirements engineering with a focus on data protection. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering* (pp. 419-424). https://doi.org/10.1145/3555228.3555262

Dąbrowski, J., Letier, E., Perini, A., & Susi, A. (2020). Mining user opinions to support requirement engineering: an empirical study. In *International Conference on Advanced Information Systems Engineering* (pp. 401–416). Springer. https://doi.org/10.1007/978-3-030-49435-3_25

Dehn, S., Jacobs, G., Zerwas, T., Berroth, J., Hötter, M., Korten, M., ... & Fleischer, D. (2023). On identifying possible artificial intelligence applications in requirements engineering processes. *Forschung im Ingenieurwesen*, 87(1), 497–506. https://doi.org/10.1007/s10010-023-00657-8

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2024). QLoRA: efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36. https://doi.org/10.48550/arXiv.2305.14314

Ding, T., Chen, T., Zhu, H., Jiang, J., Zhong, Y., Zhou, J., ... & Liang, L. (2023). The efficiency spectrum of large language models: an algorithmic survey. https://doi.org/10.48550/arXiv.2312.00678

Dubois, Y., Liang, P., & Hashimoto, T. (2024). Length-controlled AlpacaEval: a simple debiasing of automatic evaluators. In *First Conference on Language Modeling*. https://openreview.net/pdf?id=CybBmzWBX0

Franch, X., Henriksson, A., Ralyté, J., & Zdravkovic, J. (2021). Data-driven agile requirements elicitation through the lenses of situational method engineering. In *2021 IEEE 29th International Requirements Engineering Conference (RE)* (pp. 402–407). IEEE. https://doi.org/10.1109/RE51729.2021.00045

Giray, L. (2023). Prompt engineering with ChatGPT: a guide for academic writers. *Annals of Biomedical Engineering*, 51(12), 2629–2633. https://doi.org/10.1007/s10439-023-03272-4

Gu, J., Han, Z., Chen, S., Beirami, A., He, B., Zhang, G., ... & Torr, P. (2023). A systematic survey of prompt engineering on vision-language foundation models. https://doi.org/10.48550/arXiv.2307.12980

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). LoRA: low-rank adaptation of large language models. https://doi.org/10.48550/arXiv.2106.09685

Hubka, V., Andreasen, M. M., Eder, W. E., & Hills, P. J. (1988). *Practical Studies in Systematic Design*. Butterworths.

ISO 29148. (2018). Systems and Software Engineering—Life Cycle Processes—Requirements Engineering. ISO.

Jiang, Z., Xu, F. F., Araki, J., & Neubig, G. (2020). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8, 423-438. https://doi.org/10.1162/tacl_a_00324

Jin, D., Jin, Z., Chen, X., & Wang, C. (2024). MARE: multi-agents collaboration framework for requirements engineering. https://doi.org/10.48550/arXiv.2405.03256

Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., & Narayanan, A. (2024). AI agents that matter. https://doi.org/10.48550/arXiv.2407.01502.

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. *CoRR*, abs/1412.6980. https://api.semanticscholar.org/CorpusID:6628106

Kombrink, S., Mikolov, T., Karafiát, M., & Burget, L. (2011). Recurrent neural network based language modeling in meeting recognition. In *Interspeech*. https://doi.org/10.21437/Interspeech.2011-720

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., & Carlini, N. (2021). Deduplicating training data makes language models better. https://doi.org/10.48550/arXiv.2107.06499

Lim, S., Henriksson, A., & Zdravkovic, J. (2021). Data-driven requirements elicitation: a systematic literature review. *SN Computer Science*, 2(1), 16. https://doi.org/10.1007/s42979-020-00416-4

Malik, U. A., Bangash, Y. A., Iqbal, W., Zhenisbekovna, S. M., & Hammad, M. (2023). Automated conflict detection in software functional requirements using rule-based natural language processing. Preprint available at Research Square. https://doi.org/10.21203/rs.3.rs-3442526/v1

Moradi, M., Yan, K., Colwell, D., Samwald, M., & Asgari, R. (2024). Exploring the landscape of large language models: Foundations, techniques, and challenges. https://doi.org/10.48550/arXiv.2404.11973

Norheim, J. J., Rebentisch, E., Xiao, D., Draeger, L., Kerbrat, A., & de Weck, O. L. (2024). Challenges in applying large language models to requirements engineering tasks. *Design Science*, 10, e16. https://doi.org/10.1017/dsj.2024.8

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K.-H. (2007). *Engineering Design. A Systematic Approach* (3rd ed.). Springer. https://doi.org/10.1007/978-1-84628-319-2

Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., López, L., & Gorschek, T. (2021). The state-of-practice in requirements elicitation: an extended interview study at 12 companies. *Requirements Engineering*, 26, 273–299. https://doi.org/10.1007/s00766-020-00345-x

Patil, S., Joshi, P., Ingle, A., Jayappa, A., & Ketkar, O. (2023). Text extraction and finetuning transformers for abstractive summarisation. In *2023 7th International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICCUBEA58933.2023.10392203

Pokorni, B., Popescu, D., & Constantinescu, C. (2022). Design of cognitive assistance systems in manual assembly based on quality function deployment. *Applied Sciences*, 12(8), 3887. https://doi.org/10.3390/app12083887
Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkn, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730–27744. https://doi.org/10.48550/arXiv.2203.02155
Ray, A. T., Cole, B. F., Pinon Fischer, O. J., Bhat, A. P., White, R. T., & Mavris, D. N. (2023). Agile methodology for the standardization of engineering requirements using large language models. *Systems*, 11(7), 352. https://doi.org/10.3390/systems11070352
Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2024). Requirements engineering using generative AI: prompts and prompting patterns. In *Generative AI for Effective Software Development* (pp. 109–127). Springer. https://doi.org/10.1007/978-3-031-55642-5_5
Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A systematic survey of prompt engineering in large language models: techniques and applications. https://doi.org/10.48550/arXiv.2402.07927
Schlattmann, J., & Seibel, A. (2021). *Structure and Organization of Product Development Projects*. Springer. https://doi.org/10.1007/978-3-030-81046-7
Schubert, M., Riedlinger, T., Kahl, K., Kröll, D., Schoenen, S., Šegvić, S., & Rottmann, M. (2024). Identifying label errors in object detection datasets by loss inspection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (pp. 4582–4591). https://openaccess.thecvf.com/content/WACV2024/papers/Schubert_Identifying_Label_Errors_in_Object_Detection_Datasets_by_Loss_Inspection_WACV_2024_paper.pdf
Singhal, P., Goyal, T., Xu, J., & Durrett, G. (2023). A long way to go: investigating length correlations in RLHF. https://doi.org/10.48550/arXiv.2310.03716
Su, W., Tang, Y., Ai, Q., Wang, C., Wu, Z., & Liu, Y. (2024). Mitigating entity-level hallucination in large language models. https://doi.org/10.48550/arXiv.2407.09417
Sun, X., Ji, Y., Ma, B., & Li, X. (2023). A comparative study between full-parameter and LoRA-based fine-tuning on Chinese instruction data for instruction following large language model. https://doi.org/10.48550/arXiv.2304.08109.
Talebirad, Y., & Nadiri, A. (2023). Multi-agent collaboration: harnessing the power of intelligent LLM agents. https://doi.org/10.48550/arXiv.2306.03314.
Tjuatja, L., Chen, V., Wu, T., Talwalkwar, A., & Neubig, G. (2024). Do LLMs exhibit human-like response biases? A case study in survey design. *Transactions of the Association for Computational Linguistics*, 12, 1011–1026. https://doi.org/10.1162/tacl_a_00685
Van Vliet, M., Groen, E. C., Dalpiaz, F., & Brinkkemper, S. (2020). Identifying and classifying user requirements in online feedback via crowdsourcing. In *Requirements Engineering: Foundation for Software Quality* (pp. 143–159). Springer. https://doi.org/10.1007/978-3-030-44429-7_11
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. https://doi.org/10.48550/arXiv.1706.03762
VDI 2221 Part 1. (2019). *Design of Technical Products and Systems—Model of Product Design*. Beuth.
White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2024). ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative AI for Effective Software Development* (pp. 71–108). Springer. https://doi.org/10.1007/978-3-031-55642-5_4c
Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., ... & Mann, G. (2023). BloombergGPT: a large language model for finance. https://doi.org/10.48550/arXiv.2303.17564
Xie, T., Wan, Y., Huang, W., Yin, Z., Liu, Y., Wang, S., ... & Hoex, B. (2023). DARWIN series: domain specific large language models for natural science. https://doi.org/10.48550/arXiv.2308.13565
Yang, H., Liu, X. Y., & Wang, C. D. (2023). FinGPT: open-source financial large language models. https://doi.org/10.48550/arXiv.2306.06031

Appendix A

Prompt details for GPT-4

No "'markdown and'" at start and end. Create a detailed and structured requirements list that can be used for product development for [product]. Use Markup to structure the text. Do not use explanations, just provide the necessary information/facts about this list. Follow this structure:

Product design specifications document for [product]

1. Main function	
Description	Clearly define the primary purpose of the product. What is the product designed to achieve or facilitate? Specify the main functional goal.
Key features	List the key features that enable the main function. These should include any unique selling points or critical operational capabilities.
2. Functionally determined properties	
Performance requirements	Specify the essential performance metrics that the product must meet, such as power output, inputs, modes of action, speed, efficiency, and capacity.
Functionality	List all required functionalities in detail, ensuring that each function aligns with the main purpose of the product.
3. Operational properties	
Operating conditions	Detail the environmental and operational conditions under which the product is expected to perform. Be creative.
User interaction	Outline how users will interact with the product. Include user interface requirements, mechanical controls, buttons, signals, and feedback systems.
Support systems	Specify any additional systems or services needed for the product to operate effectively, such as software, network requirements, or external devices.
4. Service life and maintainability	
Expected lifespan	State the expected operational lifespan of the product under typical usage conditions in years. Provide specific time frames or cycles of use. Outline the maintenance or service intervals and the type of maintenance activities required to maximize the product's lifespan. Define the reliability requirements in quantitative terms, such as expected mean time between failures.
Common pitfalls and important product parts	Define the core product modules. Include any industry-specific testing protocols. Describe the level of fault tolerance required in specific operational usage time. Detail any redundancy or backup systems that must be in place.
End-of-life handling	Imagine an end-of-life plan for the product, including recycling, refurbishing, or disposal procedures. Include considerations for environmental impact and materials that could be recycled. Give specifications.
Service manuals	Provide comprehensive service manuals with detailed instructions on maintenance procedures, including troubleshooting guides and schematic diagrams. Give specifications.
Standardized parts	Use standardized parts where possible to minimize the variety of spare parts needed and simplify procurement. Give specifications.
5. Safety	
Fail-safe mechanisms	Design the product with auto-shutoff features that prevent or minimize risk of injury or damage in the event of a malfunction. Give specifications.
Protective features	Include necessary guards, testing methods, and warnings to protect users from potential hazards inherent in the product's operation. Give specifications.
6. Ergonomic properties	
User interface	Design interfaces that are intuitive and easy to use, with controls within easy reach and visibly labelled. Consider the use of touchscreens or voice-activated controls for enhanced usability. Give specifications.
Accessibility	Design all components to be easily accessible for maintenance and repair. Components like batteries or filters should be reachable without specialized tools. Implement a modular design that allows individual components to be replaced or upgraded independently, reducing downtime and maintenance costs. Give specifications.
Physical comfort	Ensure that any physical interaction with the product does not strain the user. If the product requires manual interaction, adjust height and angles to accommodate a wide range of users.
Visual and auditory feedback	Incorporate clear visual or auditory signals to indicate active processes or issues, enhancing user understanding and interaction. Give specifications.
Dimensions	Provide exact dimensions. Specify more space for ventilation or connections (power, cables).
7. Appearance and design	
Aesthetic design	Design a product that is visually appealing to the target market and that aligns with current design trends while remaining timeless. Consider the colour, shape, and texture that reflect the product's branding and intended market. Give specifications.
Finish and materials	Use finishes that are durable, easy to clean, and attractive. The material choices should also reflect the product's positioning in the three price categories. The user experience should ensure that the product is intuitive and easy to use. Give specifications.

Continued.

Branding	Incorporate branding elements such as logos, colour schemes and general usage to enhance the overall design so it is easily recognizable. Make specific design suggestions for user experience.
8. Law and standards	
Regulatory compliance	Identify and comply with all relevant local, national, and international regulations affecting the product, such as safety, environmental, and import/export laws. List the laws and what they mean for the product. Give specifications.
Certifications	List required certifications (e.g., CE, FCC, ISO) that the product must obtain to be sold in specific markets. Provide a plan for achieving these certifications, including testing phases and responsible parties. What does the standard do, and how does it influence product design?
Relevant patents	Ensure that the product design respects existing patents. Name all patents that have features similar to the new product. Which patents have solutions for connected problems and what aspect of the product is affected? List them.
9. Economics	
Pricing strategy	Develop a pricing strategy and give three price brackets: budget pricing, middle of the market, and premium.
Alternative revenue strategies	Give different approaches to make revenue with the product, like making the product cheap and making money with consumables. Show different ways other than selling the product with profit.

Appendix B

Output from ReqGPT for a smart electronic toothbrush (excerpt)

1. Main Function

Description

- The primary purpose of the smart electric toothbrush is to facilitate efficient and effective oral hygiene through automated brushing.
- The product is designed to enhance user experience by integrating smart features such as app connectivity, personalized brushing programs, and real-time feedback.

Key Features

- Automated Brushing: Oscillating, pulsating, or sonic brushing action to remove plaque effectively.
- App Connectivity: Bluetooth or WIFI integration for syncing brushing data with a companion app.
- Personalized Brushing Programs: Customizable brushing modes based on user needs (Sensitive Teeth, Whitening, Gum Care).
- Real-time Feedback: Vibration or audio signals to guide the user for the correct brushing technique.
- Battery Management: Rechargeable battery with smart charging indicators.
- Water Resistance: IPX7 or higher rating for waterproofness.
- Replaceable Brush Heads: Compatibility with multiple types of replaceable brush heads.
- Smart Timer: Built-in timer to ensure the recommended brushing duration (e.g., 2 minutes).
- Pressure Sensor: Alerts user when brushing too hard to prevent enamel damage.

2. Functionally Determined Properties

Performance Requirements

- Battery Life: Minimum of 14 full brushing sessions on a single charge.
- Charging Time: Less than 4 hours to fully charge.
- Brushing Modes: At least 5 distinct brushing modes (e.g., Daily Clean, Sensitive, Whitening, Gum Care).
- Oscillations Per Minute (OPM): Minimum of 31,000 OPM for sonic models.
- Pulsations Per Minute (PPM): Adjustable pulsations from 5,000 to 31,000 PPM.
- App Compatibility: Compatible with iOS and Android devices.
- Bluetooth Range: Up to 10 meters for seamless connection.
- Data Storage: Ability to store brushing history for at least 6 months.
- Waterproof Depth: IPX7 rating to withstand immersion in up to 1 meter of water. [. . .]