Prompt-with-Me: in-IDE Structured Prompt Management for LLM-Driven Software Engineering

Ziyou Li
Delft University of Technology
JetBrains Research
Delft, The Netherlands
ziyou.li@tudelft.nl

Agnia Sergeyuk
JetBrains Research
Belgrade, Serbia
agnia.sergeyuk@jetbrains.com

Maliheh Izadi
Delft University of Technology
Delft, The Netherlands
m.izadi@tudelft.nl

***Abstract*—Large Language Models are transforming software engineering, yet prompt management in practice remains ad hoc, hindering reliability, reuse, and integration into industrial workflows. We present *Prompt-with-Me*, a practical solution for structured prompt management embedded directly in the development environment. The system automatically classifies prompts using a four-dimensional taxonomy that encompasses intent, author role, software development lifecycle stage, and prompt type. To improve prompt reuse and quality, Prompt-with-Me suggests language refinements, masks sensitive information, and extracts reusable templates from a developer's prompt library.**

Our taxonomy study of 1,108 real-world prompts demonstrates that modern LLMs can accurately classify software engineering prompts. Furthermore, our user study with 11 participants shows strong developer acceptance, with high usability (Mean SUS=73), low cognitive load (Mean NASA-TLX=21), and reported gains in prompt quality and efficiency through reduced repetitive effort. Lastly, we offer actionable insights for building the next generation of prompt management and maintenance tools for software engineering workflows.

***Index Terms*—Large Language Models, Human Computer Interaction, Prompt Engineering, Software Engineering**

I. Introduction

Large Language Models (LLMs) such as GPT-4 and Claude are rapidly transforming modern software engineering (SE). Tools powered by these models have moved from experimental prototypes to integral components of everyday development workflows. For instance, industry reports on GitHub Copilot, a GPT-powered AI pair programmer, show that developers using the tool complete coding tasks up to 55% faster, while 85% report increased confidence in their code quality [1]. These numbers highlight not only the efficiency gains brought by LLMs but also their growing influence on developer experience and software quality.

Despite the transformative role of LLMs in software engineering, the primary interface for human-LLM interaction, the *prompt*, remains surprisingly informal. Prompts are typically written ad hoc, often with fragmented grammar or inconsistent wording, even though minor phrasing changes can significantly alter model behavior [2]. Unlike source code, prompts are rarely versioned, reviewed, or maintained across the software development lifecycle, leaving critical interactions with LLMs effectively unmanaged. A recent study of 243 GitHub repositories found that prompt modifications are sparsely documented and frequently introduce logical inconsistencies [3]. As LLMs become embedded in production workflows, software teams face an emerging challenge: how to systematically treat prompts as first-class software artifacts.

To address these challenges, we offer *prompt maintenance*: the ongoing curation, documentation, and evolution of a team's prompt library to support everyday software development. This is distinct from prompt *optimization*, which targets improving a single prompt for a specific output metric. Maintenance is crucial because prompts serve diverse purposes, must adapt as requirements evolve, and should remain understandable to humans. Existing tools, however, address only fragments of this need. Automated optimizers, leveraging evolutionary algorithms or reinforcement learning, can mutate prompts to improve performance [4], [5], yet often produce ambiguous text that still requires human validation [6]. Meanwhile, LLM-Ops platforms such as PromptLayer and Helicone [7], [8] operate outside the Integrated Development Environment (IDE) and are not tailored to prompts used during active software development. Existing IDE plugins offer only prompt storage [9], providing no support for quality assurance, duplicate detection, or developer comprehension. Consequently, the software engineering community currently lacks an integrated solution for maintaining programming prompts throughout the entire software development lifecycle.

In this paper, we present **Prompt-with-Me**, a JetBrains IDE plugin designed to bridge the gap in prompt maintenance for software engineering. The tool integrates prompt management directly into the developer workflow, offering a searchable prompt library enriched with automated quality and optimization features. Each new or modified prompt is automatically classified using a research-driven taxonomy that captures its *intent*, *author role*, *SDLC phase*, and prompt *type*. Prompt-with-Me also provides real-time feedback, highlighting spelling or grammar issues, potential sensitive data, and opportunities for correction. To reduce redundancy, the tool detects duplicate prompts and, when confirmed, extracts parameterized templates for reuse. All these processes are delivered within the IDE through a guided, developer-friendly interface. This allows users to accept, edit, or ignore optimization suggestions without disrupting their workflow.

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00276

To evaluate Prompt-with-Me, we formulate the following four research questions (RQs):

- **RQ1**: To what extent can LLMs reliably *categorize* prompts used in software engineering workflows?
- **RQ2**: How *usable* is Prompt-with-Me as a system for maintaining prompts in software engineering?
- **RQ3**: How *helpful* is Prompt-with-Me for users when maintaining prompts within their working environment?
- **RQ4**: What is the impact of using Prompt-with-Me on a programmer's *perceived cognitive load* during prompt maintenance?

We address RQ1 through a three-step process. First, we collected a corpus of 1,108 real-world SE prompts. Next, we derived a four-dimensional taxonomy from this corpus, grounded in existing literature and validated by SE experts [10]. Finally, we applied LLMs as classifiers using few-shot learning to automatically assign taxonomy labels, achieving substantial inter-rater agreement with human raters (Fleiss' κ=0.72).

To investigate RQ2–RQ3-RQ4, we conducted a task-based moderated user study with 11 software developers from diverse business domains. Prompt-with-Me achieved a System Usability Scale (SUS) score of 73/100, indicating good usability. Participants reported that the tool improved prompt quality, reduced repetitive effort, and simplified their workflow compared to existing practices. They further noted that adding, refining, templating, and editing prompts required only light cognitive effort. This suggests that Prompt-with-Me effectively supports prompt maintenance without imposing significant mental load.

Our work positions prompts as first-class software artifacts, opening the door to versioning, quality assurance, and CI/CD integration of prompts. Our findings also highlight opportunities for collaborative prompt repositories, refactoring strategies, and scalable prompt analytics, ultimately reducing developer effort and improving the reliability of AI-assisted software engineering. In summary, our contributions are as follows: 1) A four-dimensional taxonomy of SE prompts and an annotated dataset of 1,108 prompts. 2)Prompt-with-Me, to the best of our knowledge, the first IDE-native tool that unifies prompt storage, classification, optimization, and template extraction. 3)An empirical evaluation of LLM performance on SE prompt classification. 4) A user study on system usability, helpfulness, and cognitive load of Prompt-with-Me. 5) Actionable insights for designing future IDE-native prompt maintenance tools.

II. Related Works

The rapid advancement of LLMs has introduced a new paradigm in software engineering, where natural language intent can be translated into executable behavior [11]–[13]. This shift has spurred the development of tools and frameworks aimed at prompt management and optimization. However, most existing solutions remain disconnected from real-world software engineering practices, treating prompts as short-lived strings rather than persistent, maintainable artifacts within the developer workflow. In this section, we survey prior efforts across two key dimensions, prompt management and prompt optimization, and identify critical gaps that hinder their industrial adoption.

A. Prompt Management Tools

Artifact visibility and management are crucial in industrial settings. Software such as PromptLayer [7] and Helicone [8] provide dashboards for prompt change tracking. But they are isolated tools and remain disconnected from the SE toolset.

Some prompt management utilities live inside the IDE itself, narrowing the gap between prompt work and everyday coding. Azure Prompt Flow [14], a prompts-ops IDE plugin, integrates prompt management. However, it is targeted for LLM-application building and is not used for direct LLM interaction. Existing JetBrains AI Assistant Prompt library allows developers to store prompts they know would be reused [9] but offer no optimization mechanism or guidance. Our system bridges this tooling gap by enhancing JetBrains AI Assistant Prompt library with guided maintenance steps in the workflow that let developers keep, evolve, and review prompts along with other project artifacts. This operationalizes the methodology proposed by earlier studies and enables real-world evaluation of the methodology [15].

B. Prompt Improvement

Systems for prompt improvement often emphasizes on their ability to automate the search or generation of effective prompts for certain tasks. They treat the search for better prompts as a black-box optimization problem and apply evolutionary algorithms, Monte-Carlo tree search, Bayesian bandit, or reinforcement learning to mutate candidate prompts [4], [5], [16], [17]. However, studies also show that automated prompt optimization does not always outperform carefully crafted manual prompts [6]. In some applications, such as data labeling, automatic pipelines can be unreliable without human oversight [18]. Empirically, this is also the case for the diverse tasks in programming. These outcomes motivate the addition of human judgment to the optimization loop. Researchers have started placing users directly into the optimization loop. *Prompt Optimization with Human Feedback* frames prompt tuning as reinforcement learning with human preference signals [19]. Interactive systems, such as *Interactive Prompt Optimization*, leverage LLM for candidate prompt generation and present candidate prompts to their user [20]. Also, *Human-in-the-loop LLM-based Agents framework* puts the agent directly in collaboration with LLM-based Agents coding workflow [21]. In addition to optimizing prompts directly, a study finds that well-structured prompt templates can significantly elevate. instruction-following performance [22]. Recent work underscores the ad-hoc nature of prompt template management [23]. Prompt-with-Me fills these gaps by embedding prompt improvement procedures directly within the JetBrains IDE, supporting automated template extraction, versioning, and categorization. This integration reduces cognitive load by aligning prompt management closely with developers' existing workflows.

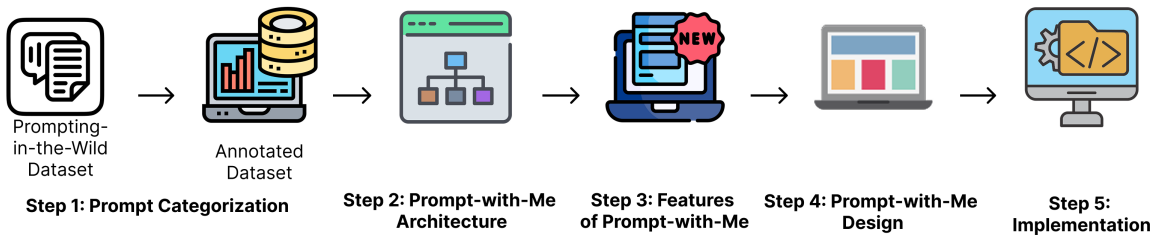

Fig. 1. Proposed Approach Structure

III. Proposed Approach

We introduce Prompt-with-Me, an IntelliJ IDEA plugin that treats prompts as first-class artifacts within the development environment. The plugin provides a structured prompt library and integrates automated tools for prompt categorization and optimization, enabling developers to manage and refine prompts directly within their project context.

Our approach begins with an empirical study of real-world prompts from software projects to derive a robust categorization scheme, which forms the foundation for clear and maintainable prompt management. Building on this categorization, we design an architecture that incorporates features to organize and optimize the prompts. We then present the core components of this architecture, followed by details of the system's design and implementation.

The remainder of this section is organized into sections corresponding to these steps: prompt categorization, system architecture, prompt processing, design, and implementation.

A. Prompt Categorization

Understanding the current state of prompts being used is crucial for building the prompt management system. And with a vision to maintain prompts in a targeted manner, such as applying a certain sequence of maintenance processes to the prompt, we need to construct a categorization of prompts used in SE. To do this, we employ the Prompting-in-the-Wild dataset [3]. The dataset provides 1262 prompt change records across 243 GitHub repositories. We employ this dataset because it is, by nature, within the context of SE, and is manually maintained by developers in sizable open-source software projects. In addition, it can be organized into per-repo prompt libraries. This aligns naturally with the prompt library form of the prompt optimization system and allows further investigation of prompt libraries.

1) Preprocessing: Upon manual inspection, we found that several prompt changes recorded in the Prompting-in-the-Wild dataset are duplicative. That is to say, we found that there are multiple instances of extremely similar changes recorded as different diffs in the dataset. While this may be vital for other studies, such as studies on duplicative commits, we only focus on tracking the prompts themselves. Therefore, deduplicating these repetitive commits is the most vital. We used Semhash, a library for text deduplication [24]. The threshold parameter for deduplication is determined to 0.999 through trial and error, through manual verification of the deduplicated prompts by the first author and by an experienced researcher in software engineering with more than 10 years of experience. The dataset size is reduced from 1262 to 1108 after deduplication. Each prompt after deduplication is also embedded using sentence embedder *ibm-granite/granite-embedding-125m-english*.

TABLE I
Single-Model Contribution to Fleiss' κ of All Models ($K_4 - K_{3,\bar{i}}$)

Category	Haiku	DS	Mistral	GPT-4o
OLD, SDLC	-0.0316	-0.0061	**+0.0359**	-0.0378
OLD, ROLE	-0.0316	**+0.0255**	+0.0240	-0.0145
OLD, INTENT	-0.0031	-0.0048	**+0.0240**	-0.0145
OLD, TYPE	+0.0325	-0.0203	**+0.0449**	-0.0519
NEW, SDLC	-0.0316	-0.0061	**+0.0360**	-0.0378
NEW, ROLE	-0.0316	**+0.0255**	+0.0240	-0.0145
NEW, INTENT	-0.0031	-0.0048	**+0.0240**	-0.0145
NEW, TYPE	+0.0325	-0.0203	**+0.0449**	-0.0519
Category Wins	0	2	**6**	0

2) Dimensions of Taxonomy: As the first step in taxonomy construction, we defined four orthogonal dimensions of taxonomy that would be of interest to the prompt optimization system, namely prompt **intent**, **SDLC** phase, prompt author **role**, and prompt **type**. Respectively, these dimensions capture the why, when, who and how of a prompt. These dimensions collectively ground the processing steps we introduce and provide avenues for future work. The intent dimension captures the prompts' underlying goal or information need. The role dimension reflects the professional role of the prompt's author in the software project (e.g., developer, data scientist, manager). The SDLC Phase dimension tags each prompt to a stage of the software development life cycle (e.g. design, implementation, testing). Finally, the dimension of prompt type denotes the style of prompt formulation (for instance, a few-shot example-based query or a template-based prompt with variables).

3) Taxonomy Construction: Next, we constructed the taxonomy in the four dimensions. We employ a method for the construction of taxonomy and the annotation of the dataset in a collaborative human-LLM manner [10]. To ensure alignment of the taxonomy content between in-company knowledge and open-source datasets, we first gathered a prompt dataset from internal projects and consented developers. Then, an initial taxonomy is constructed with the internal dataset. The taxonomy is validated by an experienced researcher in software

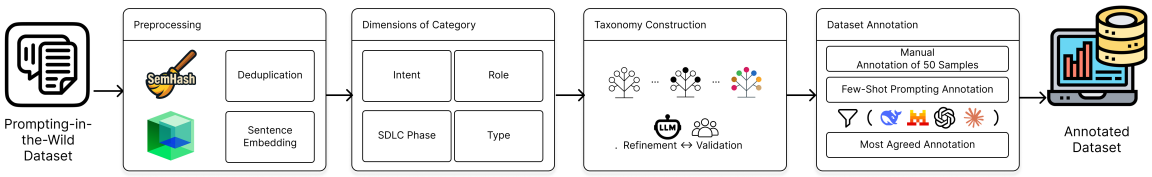

Fig. 2. Overview of Prompt Categorization

TABLE II
MISTRAL DATASET DISTRIBUTION

Category	Old Prompts	New Prompts
SDLC Phase		
General	446	444
Implementation & Coding	279	284
Testing & Quality Assurance	213	213
Planning & Design	170	167
Role		
General	686	685
Software Developer	318	316
Project Manager	68	67
Data Scientist	36	40
Intent		
Best Practices	513	517
Documentation & Explanation	269	262
Code Generation	262	266
Code Review & Analysis	64	63
Type		
Template-based	748	768
Zero-shot	205	180
Few-shot	155	160

engineering with more than 10 years of experience. Next, we refine the taxonomy in the open-source dataset by adding, removing, merging, or editing the definition of categories through the human-LLM iterative refinement loop. Finally, this loop ends with the validation of the taxonomy. The first author validates by checking the annotation of 100 random samples of the open-source dataset. If fewer than 5 prompts are incorrectly annotated, the taxonomy could be deemed valid and accurate, as provided by the literature [10]. The final taxonomy can be found in the Category column of Table II.

4) Dataset Annotation: As the data set we used is not annotated, we annotated the data set using the constructed taxonomy in Section III-A3.

The first author manually annotated a subset of prompts ($N = 50$). The remaining prompts were annotated using an optimized hybrid strategy using LLMs with enhanced few-shot prompting techniques. Each prompt for annotation employed six random manually labeled examples combined with decision trees and chain-of-thought reasoning. Each few-shot example included both old and new prompt pairs with their complete four-dimensional classifications.

To choose the lease biased annotator LLM, we follow the method of picking the model whose inclusion boosts the agreement the most, as suggested by the study [25]. To this end, we employ a *leave-one-out* strategy. First, we compute K_4, the Fleiss' κ for the four models. Next, for each model i, we calculate $K_{3,\bar{i}}$ = Fleiss' κ on the other three. Finally, we derive which model's presence causes the largest growth from $K_{3,\bar{i}}$ to K_4, which can be interpreted as the model with the most contribution, hence the best model. We experimented with the annotation on four widely-used models (GPT-4o-mini, Claude-3-Haiku, DeepSeek-Chat, and Mistral-Small) via their official APIs with default parameters for each annotation category and for both old and new prompts. The result could be inferred from Table I. We select Mistral as the annotator because it has the highest count of contributions for every combination of old/new prompt and taxonomy dimension. The distribution of classes of the resulting dataset annotated with the Mistral-Small model can be found in Table II.

B. Prompt-with-Me Architecture Overview

The Prompt-with-Me system is organized around the prompt library that stores a set of prompts and templates, as illustrated in Figure 3. The prompt library allows developers to create, edit, search, and use prompts and templates for their interaction with LLM-enabled applications. To support the refinement of artifacts in the library, we define two back-end services. The **Optimizor** monitors newly added or modified prompts and suggests optimizations to the prompts (e.g., language improvement, anonymization). Complementarily, the **Template Generator** analyses the prompts inside the library. It proposes a prompt template for the prompts to be stored in the prompt library when certain criteria are met. In our case of implementation, it is set whenever two prompts exceed a similarity score of 70%. Prompt templates are skeletons of many similar prompts, with variables for the developer to fill in for the specific tasks. Both services call external LLM APIs to perform fine-grained tasks, such as analyzing prompt wording or extracting template variable that are needed to generate their recommendations.

C. Prompt Processing Steps

Prompt-with-Me includes a suite of processing steps that effectively manage prompts in software development.

1) Classification: Prompt-with-Me annotates each newly created or modified prompt along four categorical dimensions, as defined in Section III-A2. These annotations serve as input features for downstream optimization tasks. To evaluate the feasibility of automatically classifying these dimensions, we experimented with four widely used machine learning models: Random Forest, Logistic Regression, Support Vector Machine, and a three-layer Multi-Layer Perceptron (MLP). All models were implemented using the Scikit-learn library with default hyperparameter settings and were trained and evaluated on our

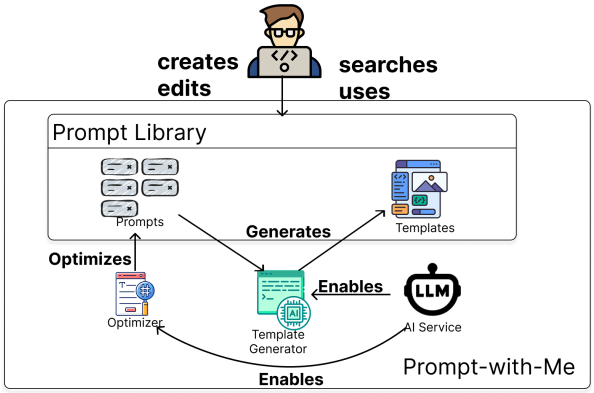

Fig. 3. Prompt-with-Me Architecture Overview

TABLE III
F1-WEIGHTED SCORES OF CLASSIFIERS ACROSS PROMPT CATEGORIZATION DIMENSIONS

Target	RF	LR	SVM	MLP
Intent	0.43	0.32	0.33	**0.45**
Type	**0.73**	0.66	0.66	0.70
Role	0.75	0.64	0.71	**0.77**
SDLC	0.44	0.40	0.40	**0.44**

annotated dataset described in Section III-A4. We report the classification performance of each model, measured by the weighted F1-score, in Table III. Among the tested models, the MLP achieved the highest overall classification performance across most dimensions. An exception was observed for prompt type classification, where the Random Forest model yielded superior accuracy. Based on these results, we adopted a hybrid approach: Random Forest is used for prompt type classification, while MLP is employed for the remaining three dimensions.

2) Similarity Detection: To maintain a concise library and encourage the reuse of prompt templates, we implemented a similarity detection system that identifies duplicate or near-duplicate prompts before they are added to the repository. This approach reduces redundancy and ensures that the collection remains both compact and versatile. To achieve robust and reliable detection, we employ a strategy that integrates complementary similarity measures:

- **Levenshtein Distance Similarity:** Calculates edit distance between prompts, normalized to a scale of 0-1.
- **Jaccard Similarity:** Compares word sets between prompts to identify common vocabulary.
- **Cosine Similarity:** Uses character n-grams to detect similar patterns in the prompt.
- Produces an ensemble similarity score using a weighted average (40% Levenshtein, 30% Jaccard, 30% Cosine), prioritizing Levenshtein to account for the frequent and semantically meaningful minor edits observed in prompt design, as supported by findings in industrial text similarity studies [26].

This ensemble score balances sensitivity to small edits with the ability to capture broader lexical and structural overlaps, thereby providing a comprehensive similarity assessment for prompt management.

3) Language Improvement: The language improvement module focuses on improving prompt clarity and correctness by performing automated spelling and grammar checks. Suggested corrections are presented to users, who can selectively apply them to refine their prompts. The system implements two key language improvements: **Spelling Correction:** Spelling errors are identified and corrected using *LanguageTool*. Corrections preserve the original case, and confidence scores are computed based on word length and the significance of the modification. **Grammar Refinement:** Grammar issues are detected with LanguageTool and presented as suggestions. These carry slightly lower confidence than spelling corrections, reflecting the inherently more subjective nature of grammar adjustments.

4) Anonymization: Prompt-with-Me includes an anonymization feature to protect sensitive information in prompts. The system uses an anonymization service with a local Name Entity Recognition (NER) model and redacts information identified as sensitive entities, such as email addresses, API keys, passwords, phone numbers, credit card numbers, IP addresses, URLs, and other personal identifiers to ensure privacy. When sensitive information is detected, the information is replaced with "[REDACTED]" placeholders. Anonymization suggestions are presented with high confidence (0.95-0.99) due to their importance for security and privacy, to raise awareness among users.

5) Template Generation: The template generation process transforms the prompts in the library into reusable templates with variable sets. This process can be triggered manually or automatically when multiple similar prompts are detected with the similarity detection feature as described in Section III-C2. The process constructs a template generation prompt, including the prompt selected for template generation, along with similar prompts in the prompt library, and the four-dimensional classification of each prompt to better capture the intent and the needs of different prompt users. The template generation prompt is sent to the LLM service to analyze text patterns. The LLM identifies common structures, variable parts, and appropriate variable names based on content analysis. The service returns a JSON response containing the generated template with placeholders (e.g., language, value), identified variables, and confidence scores for each suggestion. This approach ensures templates are contextually aware and maintain the original prompt's intent while providing flexibility through well-named variables.

6) Summarization: Prompt-with-Me provides library summarization capabilities to help users understand the overall content and patterns in their prompt collections. The summarization process analyzes the entire prompt library to extract key insights: 1) **Topic Analysis:** LLM service identifies the main topics covered by prompts in the library. 2) **Intent Distribution:** Determines the most common intent categories.

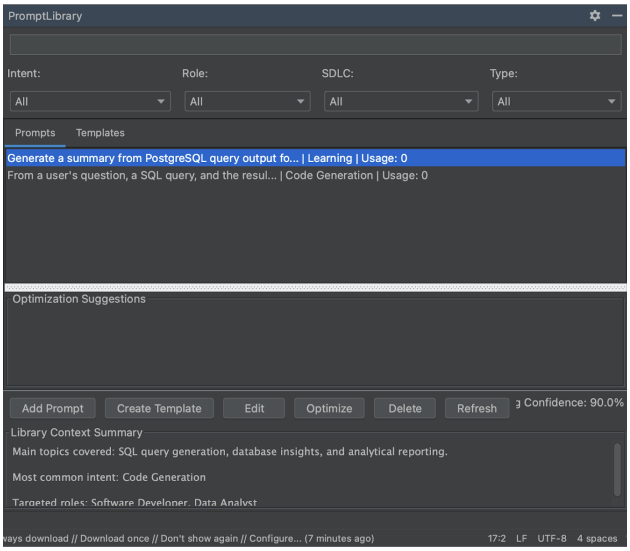

Fig. 4. Prompt-with-Me Plugin Window in JetBrains IntelliJ IDEA IDE

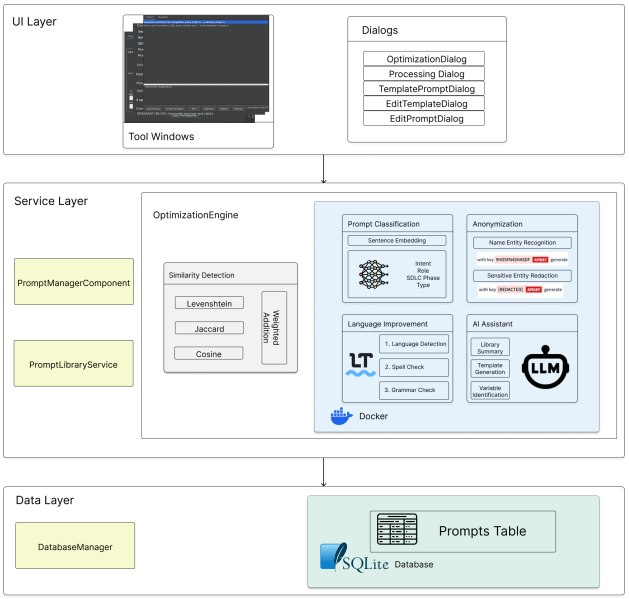

Fig. 5. Implementation Diagram

3) **Role Targeting:** Identifies which professional roles the prompts are primarily designed for. 4) **TL; DR (Short for "too long; didn't read".):** 50 to 100 words generated from the library content describing overall purpose of the library.

The summary provides a high-level overview of the prompt library's focus and usage patterns. This feature is particularly valuable for teams managing prompt libraries, as it provides quick insights without review of all prompts.

D. Prompt-with-Me Design

Prompt-with-Me is presented as a dedicated tool window, as shown in Figure 4. A filterable master list occupies the top half, while a detail pane below streams optimization suggestions for the prompts in the library. The tabs switch between prompts and templates, and the header dropdowns filter by intent, role, SDLC, and type. A toolbar offers Add, Template, Edit, Optimize, Delete, and Refresh actions. Although the current design requires further refinement, this comprehensive layout incorporates all the actions and displays required by the developer.

E. Implementation Details

The Prompt-with-Me system is implemented as a JetBrains IDE plugin using Kotlin, with a layered architecture that separates UI, service, and data concerns. An overview of the implementation can be found in Figure 5. The UI layer includes tool windows and dialogs for prompt management. The service layer contains core business logic through services. The data layer uses SQLite for local storage through a DatabaseManager. Several advanced capabilities are implemented as Docker services, including prompt classification, anonymization, language improvement, and a general AI assistant for template generation and summarization using the DeepSeek API. The plugin communicates with these services via HTTP APIs.

IV. User Study

A. Study design

Our goal for the user study is to investigate the usability, the helpfulness, and the impact on perceived cognitive load of Prompt-with-Me in a realistic SE context. To test Prompt-with-Me in a realistic SE context, we needed an initial prompt library to act as initial data in Prompt-with-Me. Therefore, we decided to choose one repository from the dataset built earlier in Section III-A4 as the prompt library for the user study case.

To find the appropriate repository in the dataset, we decided on several criteria to filter. An eligible repository should include at least two SE-related prompts (not annotated as "General") for the SDLC and role category. The repository should have at least 2 commits for prompts. In addition, there should at least be 2 files.

Among the repositories eligible for these criteria, we manually investigated the prompts and decided to use the prompts in repository *Dataherald/dataherald* for the following reasons. First, the intents of the prompts are varied, including Code Generation, Documentation & Explanation, and Learning. Second, the general scenario of the prompts are SQL querying, which is a common application scenario in organizational software development, as reported by Stack Overflow [27]. Third, the repository records explicit changes in prompt type between older and newer prompts. Specifically, prompt type change from *Zero-shot* to *Template-based*. This suggests that the prompt's capability of being modified into a template is available, which is a fit for testing the template generation ability of Prompt-with-Me.

Based on the selected prompts in the repository, we design a case study that allows testing of the main functionalities of Prompt-with-Me. The case study replicates a realistic software

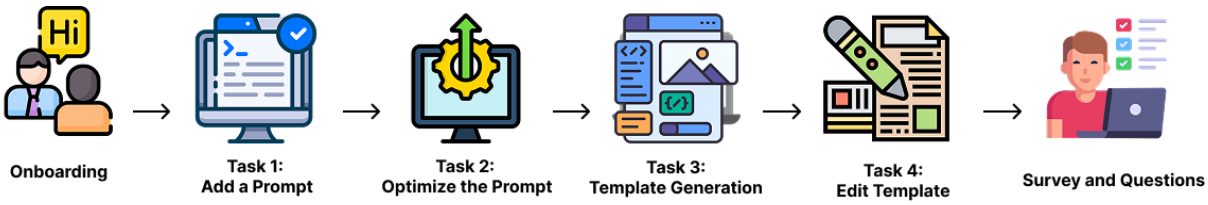

Fig. 6. User Study Workflow

development context where participants are asked to manage prompts used in LLM-assisted SQL querying tasks.

The study followed these steps:**(1) Prelude:** We first introduced the study's goals and procedures, obtained informed consent, and asked participants background questions about their occupation, coding experience, and how often they use LLMs at work. **(2) Onboarding:** The participants are asked to assume the role of a software developer working at a data analytics company. Their primary tasks involve writing, editing, and optimizing prompts that are used to interact with LLMs to generate or summarize SQL queries. Although the technical requirement for tasks is intentionally abstracted away, the scenario remains grounded in tasks that reflect actual prompt maintenance behaviors in enterprise workflows. **(3) Main Study:** Participants start JetBrains IntelliJ IDEA IDE with the Prompt-with-Me plugin, in which a prompt library containing two pre-existing prompts related to SQL summarization and response generation exists. They are then asked to perform the four tasks using Prompt-with-Me. First, they add a new prompt to the library and observe how the system gives basic properties of the prompts (e.g., length, word count), automatically classifies it in the four dimensions defined in Section III-A2, and suggests optimizations suitable for the prompt. Next, they initiate the optimization process for this prompt, accepting suggestions related to language improvement and anonymization. In the third task, participants are asked to generate a prompt template from the prompt added in the first task. They are requested to verify whether the automatically extracted variables for the template are reasonable. Finally, they edit the generated template to customize it with additional variables and variable sets. **(4) Post-Study Survey and Feedback Collection** After the main study, the participants are asked to evaluate the usability of the system by completing the standard SUS questions [28] and to evaluate the perceived cognitive load with NASA Task Load Index [29]. Raw NASA Task Load Index is selected as suggested by previous research for early studies [30]. They were also asked the following open questions: (1) Did Prompt-with-Me help you write prompts more efficiently? If yes, please elaborate how. (2) What feature(s) or aspect(s) did you find most useful in Prompt-with-Me? (3) Do you have any suggestions for making Prompt-with-Me more suitable for your needs at work?

During the survey, participants were encouraged to think aloud, and the surveyor took notes and used a transcriber to capture the words of the participants when completing the tasks. Upon completing the survey, the surveyor asked participants to additionally provide some last comments about the workflow, the task, and our system in relevance of the respective experience of the participants at work.

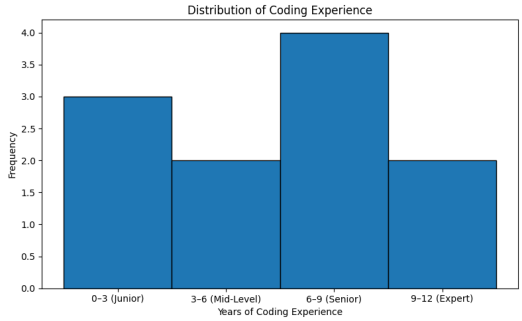

Fig. 7. User Research Panel's Coding Experience (Years), Histogram

B. Participants

The targeted user group of Prompt-with-Me is a group of software professionals working on software projects using JetBrains IDEs. They interact with LLM-powered tools in their work and would benefit from maintaining a shared prompt library, especially in settings where tasks have repeatable structures with minor variations. This situation with significant context reliance makes prompt maintenance valuable for reuse and adaptation.

To reflect the targeted user group, we recruited 11 participants with diverse professional backgrounds and coding experience. The participants are recruited through the authors' networks and social media posts. Our participants came from diverse professional sectors, including academia, finance, automotive, telecommunication and civil construction. The organization sizes vary from small companies with 20-50 employees to large organizations with 300,000+ employees. Their residence countries include The Netherlands, France, Germany, China and Japan. All reported using IDEs and prompt-based LLM applications regularly in their work and had between 1 and 10 years of coding experience. The distribution of participants' coding experience can be found from Figure 7.

C. Data Collection

We employed a usability-testing protocol suitable for both in-person and remote studies [31]. The plugin and JetBrains

TABLE IV
FLEISS' KAPPA INTER-RATER AGREEMENT FOR OLD AND NEW PROMPT CATEGORIES

Category	Fleiss' Kappa	Agreement Level [32]
OLD SDLC	0.4315	Moderate
OLD ROLE	0.6898	Substantial
OLD INTENT	0.6253	Substantial
OLD TYPE	0.5573	Moderate
TOTAL (OLD Categories)	**0.7219**	**Substantial**
NEW SDLC	0.4433	Moderate
NEW ROLE	0.6909	Substantial
NEW INTENT	0.6405	Substantial
NEW TYPE	0.5377	Moderate
TOTAL (NEW Categories)	**0.7237**	**Substantial**

IntelliJ IDEA IDE were run on the surveyor's laptop, with offline participants interacting directly and online participants gaining remote control via Zoom screen-sharing. At the beginning of the study, the moderator briefed each participant on the plugin, the goal of the study, and the tasks to be performed. With consent for data collection obtained, the moderator also encouraged the participants to think aloud to capture real-time reasoning. Throughout the tasks the moderator remained present with minimal intervention while still providing clarifications when requested. Immediately after task completion, participants filled out the SUS and raw NASA Task Load Index (NASA-TLX) survey on Google Forms.

V. RESULTS

A. RQ1: LLM Prompt Categorization

Inter-rater agreement among the four LLM annotators for prompt categorization was substantial for both old and new prompts (overall $\kappa = 0.72$; Table IV). Among the individual categories, Role labels achieved the highest consistency ($\kappa = 0.69$ for both old and new prompts), closely followed by Intent ($\kappa = 0.63$ and 0.64). Agreement on Prompt Type was moderate ($\kappa = 0.56$ and 0.54), while SDLC Phase exhibited the lowest yet still moderate reliability ($\kappa = 0.43$ and 0.44). We find that the annotation approach is less effective on SDLC phase, as Claude-3-Haiku heavily favors "Planning & Design", DeepSeek-Chat and OpenAI favors "General", each annotated more than half of all promps in their respective favored category, which Mistral showed a more balanced distribution. According to the interpretation scale proposed by Landis and Koch [32], these results suggest that current LLMs can classify software engineering prompts with moderate to substantial consistency.

B. RQ2: Usability and User Experience

Participants generally perceived the system as highly usable, with a mean SUS score of 72.73 out of 100 [CI 95% (61.54, 83.91)]. According to the interpretation guidelines proposed by Lewis [33], this score corresponds to a "good" user experience, suggesting that the system meets the usability expectations of its target audience.

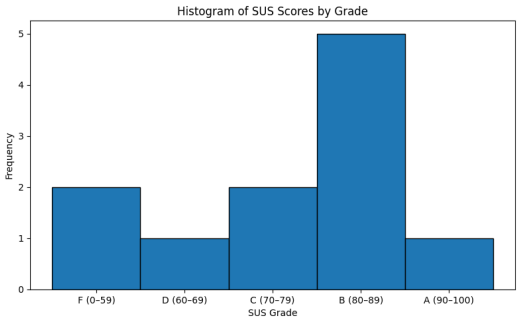

Fig. 8. Histogram of overall SUS Scores by Approximate Grading [33]

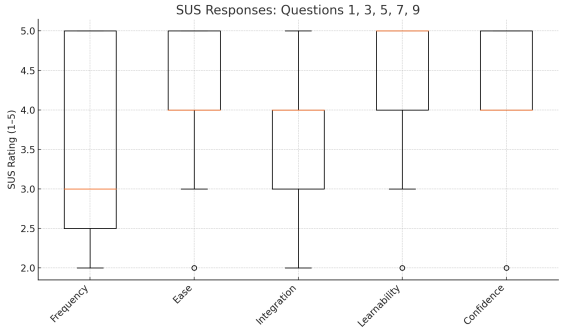

Fig. 9. Score Distribution for SUS question 1, 3, 5, 7, 9 (Positive Usability)

Figure 9 presents the participants' ratings across the evaluated usability aspects, where higher values indicate better perceived usability. Overall, the system received strong ratings for ease of use, learnability, and user confidence, with nearly all participants scoring these aspects highly. Integration also received favorable evaluations; however, it exhibits the widest distribution of ratings, suggesting that while the components work together reasonably well, there is noticeable room for improvement in their integration. The most divergent opinions concern the expected frequency of tool usage, with responses spanning the full 2–5 range and clustering around a median of 3. This pattern reflects a mixed perception: some developers

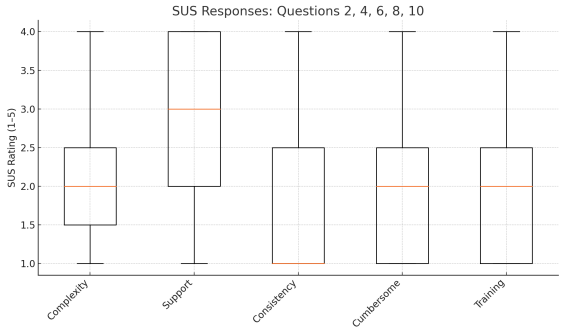

Fig. 10. Score Distribution for SUS question 2, 4, 6, 8, 10 (Negative Usability)

envision the tool as part of their daily workflow, whereas others anticipate using it only occasionally. Notably, no participant indicated that they would completely avoid using the tool. This highlights at least some perceived utility for all users.

Figure 10 presents participants' ratings for each usability question, where higher values indicate poorer usability. Four items, complexity, consistency, cumbersome usage, and training requirements, show median ratings of 2 or lower, suggesting that most participants disagreed with statements implying that Prompt-with-Me is overly complex, inconsistent, awkward, or difficult to learn. The only notable exception is the "support" item, which reflects mixed opinions regarding the need for additional guidance. Across all items, boxplots span approximately two scale points, with whiskers reaching both extremes, indicating some variability among responses. Overall, these results suggest that while the majority of participants experienced smooth and intuitive interaction, a small subset still perceived notable usability challenges.

In general, the distribution suggests generally positive usability, with some variability in anticipated usage frequency and an indication that extra support may be required.

C. RQ3: Helpfulness of Prompt-with-Me

We assessed helpfulness through participants' open-ended survey responses and reflections during the study tasks. All eleven participants reported that Prompt-with-Me accelerated their prompt-writing process, primarily due to its **template generation** feature.

Seven participants highlighted that it eliminated the need to copy, paste, or re-type similar prompts repeatedly. As Participant #10 explained, "*A lot of my prompts are almost the same, so typing them out each time is tedious. The system let me pick from my past prompts and reminded me of the exact wording I used before. It saved me a ton of time.*"

Other features, such as **automatic classification, anonymization, grammar correction, and similarity checking** were also noted as time-savers. Several participants estimated that the integrated IDE plugin saved "*a few minutes per prompt*", which accumulates over a full day of coding. For example, Participant #6 shared, "*When I am scripting in Revit's Dynamo, I hit the same scenarios across different maquettes. Being able to pull up and reuse the right prompts speeds up programming across all my 3D models.*"

Overall, participants agreed that the current version of Prompt-with-Me not only reduces repetitive work but also improves the quality of prompts.

D. RQ4: Perceived Cognitive Load

Figure 11 presents the six raw NASA-TLX subscale scores on a 0–100 scale, and Table V reports their means and 95% confidence intervals. Across all subscales, the mean scores fall within the medium workload range (10–29) defined in Table VI. Mental demand was the highest reported factor [Mean=29.55, CI 95% (14.55, 44.54)], approaching the threshold for "somewhat high", whereas all other subscales remained solidly in the medium range. The overall workload score was

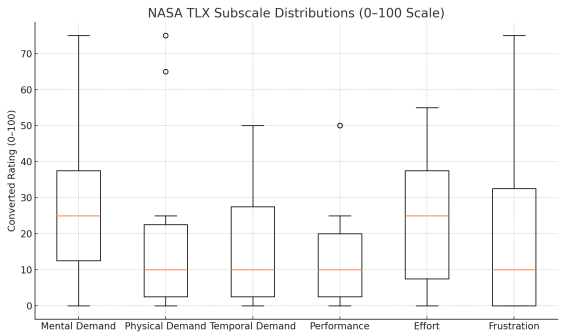

Fig. 11. NASA TLX Subscale Distributions

TABLE V
RAW TLX SCORE REPORTED BY USER STUDY PARTICIPANTS

Dimension	Mean (0–100)	95 % CI
Mental demand	29.55	[14.55, 44.54]
Physical demand	20.45	[3.98, 36.93]
Temporal demand	17.27	[5.02, 29.52]
Performance	15.91	[4.21, 27.61]
Effort	23.64	[10.92, 36.35]
Frustration	19.55	[3.38, 35.71]
Overall	**21.06**	**[9.90, 32.23]**

21.06, [CI 95% (14.55, 44.54)]. This ensures that Prompt-with-Me imposed only a **light cognitive load** on participants.

The boxplots in Figure 11 reveal long whiskers and several outliers, particularly for mental/physical demand and frustration, suggesting that few participants experienced brief periods of higher cognitive load. Nevertheless, the overall distribution patterns confirm that interacting with Prompt-with-Me generally imposed only **modest cognitive effort**.

VI. DISCUSSION

A. Insights from the Controlled Evaluation

Template Extraction Enables Prompt Reuse: Prompt-with-Me consolidates similar prompts into a single, parameterized template to enable systematic reuse and adaptation. These templates function as human-readable programs: they combine a stable core corpus with variable components that can be adjusted to different application scenarios. In our user study, a majority of participants (7 out of 11) identified template generation as the most valuable feature of Prompt-with-Me. Beyond improving efficiency, templating supports versioning and reviewing prompts as stable, reusable artifacts within

TABLE VI
INTERPRETATION BANDS FOR OVERALL NASA TLX SCORES (0–100) [34]

TLX Score (0–100)	**Interpretation**
0–9	Low workload
10–29	Medium workload
30–49	Somewhat high workload
50–79	High workload
80–100	Very high workload

a software project. This approach also has the potential to reduce errors commonly introduced through ad hoc copy-and-paste prompt editing [35], resulting in both consistency and reliability in LLM-based workflows. **Tool Adaptiveness Drives Adoption More Than Low Cognitive Load**: Although participants consistently reported a low cognitive load when using the tool, their anticipated frequency of use varied considerably. This suggests that ease of use alone does not guarantee frequent adoption. Participants from various sectors emphasized that perceived usefulness depends on how well Prompt-with-Me adapts to the specific workflows and practices of their domain. Consequently, the ability of the tool to integrate into established routines and support customization of existing workflows emerges as a factor for adoption and a worthy avenue for future work.

B. Lessons Learned for Future Designs

Our study revealed several design lessons that tool builders should consider to enhance usability and user trust. 1) **Transparency and Reversibility to Enhance User Trust**: First, participants consistently emphasized that automation is only welcomed when its effects are both transparent and reversible. Users want to understand what the tool has changed and feel confident that they can easily undo or adjust these modifications. This finding highlights the importance of providing clear, accessible explanations of any optimizations or transformations performed by the tool, along with straightforward mechanisms for reverting them. 2) **Need for Domain-Specific Customization in Taxonomy Design**: Although participants from diverse backgrounds generally accepted the four-dimensional taxonomy, users from more specialized domains expressed a need for a more customizable taxonomy that aligns closely with their specific workflows. This highlights the importance of developing extendable classification engines capable of accommodating domain-specific requirements to ensure that the system can adapt to the nuanced practices of different user groups. 3) **Simplify UI**: Some participants noted that the main window of the all-in-one plugin could benefit from a simpler, more streamlined design. While the current interface effectively presents all the information required for prompt optimization, it also contributes to an already crowded IDE environment. Beyond functional improvements, participants suggested that greater attention should be paid to the visual and interaction design to enhance usability and reduce cognitive load. 4) **Balance Privacy and Performance**: Participants from privacy-sensitive organizations preferred *local* processing for all prompt optimizations, while others prioritized *speed* and *performance* over data locality. This contrast suggests that providing a flexible option, such as a toggle between online and offline modes, could accommodate both needs and potentially broaden adoption. 5) **Context-Aware Optimizations Increase Value**: Beyond deployment preferences, participants noted that prompt optimizations could be enhanced using *IDE context* and project-specific information. 6) **Collaboration Features Enable Teamwide Benefits**: Several participants highlighted the need for *collaborative features*. Suggested capabilities included a shared prompt library with version control, in-line comments, and review mechanisms, enabling teams to build and refine prompts collectively.

C. Threats to Validity

Internal validity: Our user study involved eleven participants, which, while sufficient for a qualitative and exploratory evaluation, may not fully capture the range of developer experiences. Additionally, usefulness and mental effort were assessed through self-reported questionnaires. Although subjective measures are a standard first step in evaluating developer tools, future studies can complement them with objective metrics such as task completion time, TLX scores, or usage logs to strengthen reliability. **External validity**: Our study focused on SQL-based tasks within JetBrains IntelliJ using controlled scenarios rather than live production environments. While this allowed us to ensure consistency and isolate the effects of Prompt-with-Me, it narrows generalizability to other IDEs or workflow contexts. Expanding future evaluations to larger and more diverse participant pools will help confirm and broaden our conclusions. **Construct validity**: We operationalized usefulness and mental effort through questionnaire responses and inferred template reuse from self-reports rather than direct behavioral or long-term adoption data. While sufficient for a first investigation, future work can strengthen construct validity by combining subjective responses with behavioral logs, longitudinal studies, and productivity indicators.

VII. Conclusion

Prompts are now critical in LLM-enabled software development, yet they are often crafted and managed in an ad hoc manner. Prompt-with-Me addresses this gap by demonstrating how prompts can be systematically organized, maintained, and leveraged within development workflows. Our taxonomy study reveals that modern LLMs can reliably classify software engineering prompts, providing a foundation for structured prompt management. Complementing this, our user study shows that developers appreciate Prompt-with-Me for its usability, efficiency, and low cognitive overhead. Looking ahead, we plan to enrich the taxonomy, introduce collaborative features, and refine the plugin's interface. By integrating prompt maintenance into everyday programming tools, Prompt-with-Me takes a meaningful step toward safer, more sustainable, and developer-friendly LLM-based software development.

Acknowledgments

This work was conducted as part of the AI for Software Engineering (AI4SE) collaboration between JetBrains and Delft University of Technology. The authors acknowledge the financial support provided by JetBrains, which made this research possible. We also thank our colleagues at JetBrains who contributed to the early ideation of the *Prompt With Me* feature during internal innovation initiatives. Their input helped shape the direction, scope, and positioning of the feature. In particular, we thank Sergey Boytsov, Dariia Karaeva, Ekaterina Koshchenko, Anna Maltseva, Vladimir Poliakov, and Ilya Zaharov.

References

[1] Y. Gao and GitHub Customer Research, "Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture." https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/, May 2024.

[2] A. Chatterjee, H. S. V. N. S. K. Renduchintala, S. Bhatia, and T. Chakraborty, "POSIX: A Prompt Sensitivity Index For Large Language Models," Oct. 2024. arXiv:2410.02185 [cs].

[3] M. Tafreshipour, A. Imani, E. Huang, E. Almeida, T. Zimmermann, and I. Ahmed, "Prompting in the Wild: An Empirical Study of Prompt Evolution in Software Repositories," Jan. 2025. arXiv:2412.17298 [cs].

[4] B. Wen, P. Ke, Y. Sun, C. Wang, X. Gu, J. Zhou, J. Tang, H. Wang, and M. Huang, "HPSS: Heuristic Prompting Strategy Search for LLM Evaluators," May 2025. arXiv:2502.13031 [cs].

[5] E. Agarwal, J. Singh, V. Dani, R. Magazine, T. Ganu, and A. Nambi, "PromptWizard: Task-Aware Prompt Optimization Framework," Oct. 2024. arXiv:2405.18369 [cs].

[6] Y. Zhou, Y. Zhao, I. Shumailov, R. Mullins, and Y. Gal, "Revisiting Automated Prompting: Are We Actually Doing Better?," June 2023. arXiv:2304.03609 [cs].

[7] MagnivOrg, "PromptLayer - Maintain a log of your prompts and OpenAI API requests. Track, debug, and replay old completions.," July 2025. https://github.com/MagnivOrg/prompt-layer-library.

[8] Helicone, "Helicone: Open-source observability platform for large language model apis," July 2025. https://github.com/Helicone/helicone.

[9] JetBrains, "Prompt Library | AI Assistant," June 2025. https://www.jetbrains.com/help/ai-assistant/settings-reference-prompt-library.html.

[10] C. Shah, R. W. White, R. Andersen, G. Buscher, S. Counts, S. Snigdha, S. Das, A. Montazer, S. Manivannan, J. Neville, X. Ni, T. Safavi, S. Suri, M. Wan, L. Wang, and L. Yang, "Using Large Language Models to Generate, Validate, and Apply User Intent Taxonomies," 2023.

[11] A. Deljouyi, R. Koohestani, M. Izadi, and A. Zaidman, "Leveraging large language models for enhancing the understandability of generated unit tests," in *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, pp. 1–13, 2025.

[12] M. Izadi, J. Katzy, T. Van Dam, M. Otten, R. M. Popescu, and A. Van Deursen, "Language Models for Code Completion: A Practical Evaluation," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, (New York, NY, USA), pp. 1–13, Association for Computing Machinery, Apr. 2024.

[13] A. C. Ionescu, S. Titov, and M. Izadi, "A multi-agent onboarding assistant based on large language models, retrieval augmented generation, and chain-of-thought," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 1208–1212, 2025.

[14] Microsoft, "Prompt flow in Azure AI Foundry portal - Azure AI Foundry," June 2025. https://learn.microsoft.com/en-us/azure/ai-foundry/concepts/prompt-flow.

[15] Z. Chen, C. Wang, W. Sun, G. Yang, X. Liu, J. M. Zhang, and Y. Liu, "Promptware Engineering: Software Engineering for LLM Prompt Development," Mar. 2025. arXiv:2503.02400 [cs].

[16] R. Murthy, M. Zhu, L. Yang, J. Qiu, J. Tan, S. Heinecke, C. Xiong, S. Savarese, and H. Wang, "Promptomatix: An Automatic Prompt Optimization Framework for Large Language Models," July 2025. arXiv:2507.14241 [cs].

[17] Z. Wu, X. Lin, Z. Dai, W. Hu, Y. Shu, S.-K. Ng, P. Jaillet, and B. K. H. Low, "Prompt Optimization with EASE? Efficient Ordering-aware Automated Selection of Exemplars," Oct. 2024. arXiv:2405.16122 [cs].

[18] Z. He, S. Naphade, and T.-H. K. Huang, "Prompting in the Dark: Assessing Human Performance in Prompt Engineering for Data Labeling When Gold Labels Are Absent," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, (Yokohama Japan), pp. 1–33, ACM, Apr. 2025.

[19] X. Lin, Z. Dai, A. Verma, S.-K. Ng, P. Jaillet, and B. K. H. Low, "Prompt Optimization with Human Feedback," May 2024. arXiv:2405.17346 [cs].

[20] J. Li and R. Klinger, "iPrOp: Interactive Prompt Optimization for Large Language Models with a Human in the Loop," June 2025. arXiv:2412.12644 [cs].

[21] W. Takerngsaksiri, J. Pasuksmit, P. Thongtanunam, C. Tantithamthavorn, R. Zhang, F. Jiang, J. Li, E. Cook, K. Chen, and M. Wu, "Human-In-the-Loop Software Development Agents," Jan. 2025. arXiv:2411.12924 [cs].

[22] P. Liu, H. Wang, C. Zheng, and Y. Zhang, "Prompt Fix: Vulnerability Automatic Repair Technology Based on Prompt Engineering," in *2024 International Conference on Computing, Networking and Communications (ICNC)*, pp. 116–120, IEEE Computer Society, Feb. 2024.

[23] Y. Mao, J. He, and C. Chen, "From Prompts to Templates: A Systematic Prompt Template Analysis for Real-world LLMapps," Apr. 2025. arXiv:2504.02052 [cs].

[24] Thomas van Dongen and Stephan Tulkens, "SemHash: Fast Semantic Text Deduplication & Filtering," Apr. 2025. GitHub repository. Available: https://github.com/MinishLab/semhash.

[25] S. P. M. Davoudi, A. G. Davodi, A. Amiri-Margavi, and M. Jafari, "Collective Reasoning Among LLMs: A Framework for Answer Validation Without Ground Truth," June 2025. arXiv:2502.20758 [stat].

[26] W. Andrzejewski, B. Bębel, P. Boiński, M. Sienkiewicz, and R. Wrembel, "Text similarity measures in a data deduplication pipeline for customers records," *Proceedings of the 25th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*, 2023.

[27] Stack Overflow, "Stack Overflow Developer Survey 2023," Aug. 2023.

[28] J. Brooke, "Sus: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, CRC Press, 1st edition ed., 1996. Available: https://www-taylorfrancis-com.tudelft.idm.oclc.org/chapters/edit/10.1201/9781498710411-35/sus-quick-dirty-usability-scale-john-brooke.

[29] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," in *Advances in Psychology* (P. A. Hancock and N. Meshkati, eds.), vol. 52 of *Human Mental Workload*, pp. 139–183, North-Holland, Jan. 1988.

[30] E. A. Bustamante and R. D. Spain, "Measurement Invariance of the Nasa TLX," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, pp. 1522–1526, Sept. 2008. Publisher: SAGE Publications Inc.

[31] A. Boswell, "Moderated usability testing," May 2025. Lyssna blog. Available: https://www.lyssna.com/blog/moderated-usability-testing/.

[32] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. Publisher: International Biometric Society.

[33] James R. (Jim) Lewis, Ph.D and Jeff Sauro, PhD, "Item Benchmarks for the System Usability Scale - JUX," *JUX - The Journal of User Experience*, May 2018.

[34] M. Brillinger, S. Manfredi, D. Leder, M. Bloder, M. Jäger, K. Diwold, A. Kajmakovic, M. Haslgrübler, R. Pichler, M. Brunner, S. Mehr, and V. Malisa, "Physiological workload assessment for highly flexible fine-motory assembly tasks using machine learning," *Computers & Industrial Engineering*, vol. 188, p. 109859, Feb. 2024.

[35] J. T. Liang, M. Lin, N. Rao, and B. A. Myers, "Prompts Are Programs Too! Understanding How Developers Build Software Containing Prompts," Apr. 2025. arXiv:2409.12447 [cs].