IPTEK The Journal for Technology and Science, 37(2), 2026 (e/pISSN:2088-2033, 0853-4098)

DOI: 10.12962/j20882033.v37i2.9252 | Received 09 February, 2026; Revised 24 June, 2026; Accepted 02 July, 2026

ORIGINAL RESEARCH

Implementation of Chatbot for Generating Natural Language to SQL Queries

Rahmad Santosa[1] | Hartantya Ainiyatuts Tsaniyah[1] | Yoga Ari Tofan*[1] | Adetiya Bagus Nusantara[1] | Radityo Prasetianto Wibowo[2]

[1]Dir. of Technology and Information System Development, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia

[2]Dept. of Information System, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia

Correspondence

*Email: y.aritofan@gmail.com

Present Address

Gedung Research Center, Jl. Teknik Kimia, Kampus ITS Sukolilo, Surabaya 60111, Indonesia

Abstract

This paper presents the practical implementation of a chatbot system designed to bridge natural language communication with relational database interactions by automatically generating Structured Query Language (SQL) queries. The system leverages the OpenAI API — specifically the GPT-3.5 chat completion model — to interpret user requests expressed in natural language and transform them into syntactically correct and semantically meaningful SQL commands compatible with Microsoft SQL Server. We describe the system architecture, the construction of a domain-specific data catalog that serves as contextual knowledge for the model, and a few-shot prompting strategy using role-based prompt design. To evaluate the system, we recruited eight respondents divided into four groups based on their SQL proficiency levels and observed their performance on eighteen query tasks of varying complexity, both with and without chatbot assistance. Results show that chatbot assistance substantially reduced task completion time and improved success rates for respondents with limited SQL experience, while offering marginal benefit to expert users. We also identify key limitations of the current approach, including non-deterministic outputs and failures on queries requiring PIVOT operations, and discuss directions for mitigation.

KEYWORDS:

chatbot, few-shot prompting, natural language to SQL, NL2SQL, openAI GPT-3.5, SQL server

1 | INTRODUCTION

The integration of natural language processing (NLP) with relational database systems has attracted considerable research interest, driven by the practical need to make data more accessible to users who lack expertise in Structured Query Language (SQL). Translating natural language queries into SQL commands, commonly referred to as *Natural Language to SQL* (NL2SQL) or

Text-to-SQL, removes a significant barrier between end users and the information stored in databases. This paper describes the implementation of a chatbot system for NL2SQL query generation, built on top of the OpenAI GPT-3.5 API using a chat completion architecture. Rather than training a model from scratch, our approach relies on *few-shot prompting*: we supply the model with a structured data catalog describing the target database schema, a set of system-level constraints, and a small number of example query-answer pairs. This design makes the system practical to deploy without specialized machine learning infrastructure. The primary objective of this study is to evaluate how effectively a GPT-3.5-based chatbot can assist users with varying levels of SQL proficiency in retrieving data from a university accreditation database. We focus on a real-world institutional setting at Institut Teknologi Sepuluh Nopember (ITS), Surabaya, where staff members regularly receive informal data requests that require translating loosely worded questions into complex SQL queries against the university's internal database systems. The remainder of this paper is organized as follows. Section 2 reviews related work on NL2SQL approaches. Section 3 describes the system architecture, data catalog construction, prompt design, and experimental setup. Section 4 presents the results and discusses the findings, including observed failure modes. Finally, Section 5 concludes the paper and outlines directions for future work.

2 | PREVIOUS RESEARCH

The translation of natural language into SQL queries, commonly referred to as *Natural Language to SQL* (NL2SQL) or *Text-to-SQL*, has been an active area of research for several decades[1]. Early approaches were primarily rule-based, relying on handcrafted grammar and semantic rules to map natural language expressions to database-specific SQL syntax[2–6]. These systems achieved high accuracy on narrow, well-defined domains but struggled to generalize across different database schemas and query types. Popescu *et al.* proposed a theoretical framework for reliable natural language interfaces to databases (NLIDB) and demonstrated that over 80% of semantically tractable questions could be mapped correctly to SQL[7]. Similarly, Singh and Solanki developed a rule-based interface using semantic matching and production rules, converting natural language queries through tokenization and part-of-speech tagging steps[8]. With the rise of deep learning, neural approaches have largely superseded rule-based methods due to their ability to generalize without extensive manual engineering. Xu *et al.* introduced SQLNet, a sketch-based model that bypasses the sequence-to-sequence "order-matters" problem by predicting SQL components using a dependency graph and column attention mechanism[9]. SQLNet outperformed prior state-of-the-art methods by 9–13% on the WikiSQL benchmark. Building on this, Yu *et al.* proposed SyntaxSQLNet, a syntax tree-based decoder for complex and cross-domain queries, achieving a 9.5% improvement in exact matching accuracy over previous work on the Spider dataset[10]. Comprehensive surveys by Katsogiannis-Meimarakis and Koutrika[11] and Qin *et al.*[12] provide a detailed taxonomy of neural Text-to-SQL systems, covering encoder design, schema linking, and decoder strategies. These studies highlight that complex multi-table queries with nested structures remain an open challenge for deep learning models. More recently, large language models (LLMs) have redefined the state of the art in NL2SQL tasks. Liu *et al.*[13] conducted the first comprehensive evaluation of ChatGPT's Text-to-SQL capability across twelve benchmark datasets in a zero-shot setting. Their results showed that although ChatGPT does not consistently outperform specialized fine-tuned models on complex benchmarks, it demonstrates strong performance without any task-specific training. Notably, ChatGPT exceeded the fine-tuned state-of-the-art model by 4.1% in the ADVETA scenario. Research on few-shot prompting strategies has shown further improvements. Nan *et al.*[14] demonstrated that combining diversity and similarity in demonstration selection, together with database-relevant knowledge augmentation, improved execution accuracy on the Spider benchmark by 2.5 percentage points compared to previous state-of-the-art systems. A broader survey by Liu *et al.*[15] reviews Text-to-SQL research in the LLM era, covering model architectures, prompting strategies, evaluation methods, and common error patterns in deployed systems. An important point of comparison for the present work is the application of NL2SQL techniques to domain-specific databases. Kurniawan *et al.*[16] developed a chatbot using a rule-based semantic parsing approach for an academic information system at Sriwijaya University in Indonesia, achieving 96.72% accuracy on 122 test queries. However, the rule-based architecture required extensive manual configuration of grammar and semantic rules. Kanburoğlu and Tek[17] developed TUR2SQL, a cross-domain Turkish Text-to-SQL dataset, and compared SQLNet with ChatGPT. Their findings showed that ChatGPT generated more accurate SQL queries from Turkish natural language inputs. ChatGPT has also been explored in related educational applications, including the automatic generation and evaluation of online multiple-choice tests, reducing the time required for test creation and assessment[18]. The present study differs from prior work in several important respects. First, it targets a proprietary, domain-specific university accreditation database that operates primarily in the Indonesian language. Second, it utilizes OpenAI's GPT-3.5 chat completion API with a few-shot prompting strategy rather than a specialized fine-tuned Text-to-SQL model. Finally, instead of focusing on benchmark

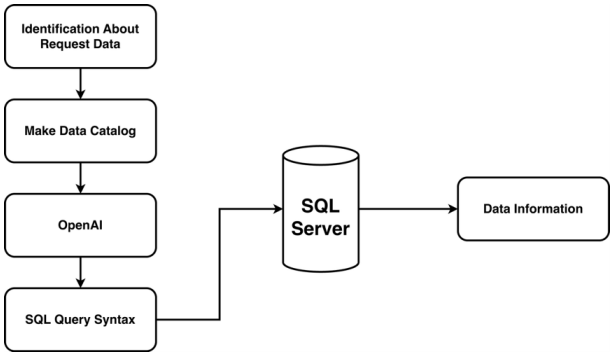

FIGURE 1 Flow SQL generator with OpenAI.

datasets such as WikiSQL or Spider, this research evaluates the system through a human-subject study. The evaluation measures the impact of chatbot assistance on query completion time and query accuracy across participants with varying levels of SQL proficiency. This user-centric evaluation provides practical insights into real-world adoption that benchmark-based evaluations alone cannot fully capture.

3 | METHOD

This section describes the four main stages of the proposed system: (1) identification of frequent data requests, (2) construction of a data catalog, (3) integration of the OpenAI GPT-3.5 API for SQL query generation, and (4) execution and validation on Microsoft SQL Server. The overall system flow is illustrated in Figure 1 .

3.1 | Identification of Data Requests

To understand the types of queries that the system must handle, we first conducted interviews with two staff members at ITS who routinely receive informal data requests from other departments. This process produced 27 distinct data requests, which were recorded in their original form—often written as informal messages in Indonesian—and then normalized into structured query descriptions. A sample of these requests is shown in Table 1 . Each entry in the dataset includes five fields: the original message text, a normalized version of the request, the topic of the query, the data group (lecturer, student, or administrative staff), and whether the query is relevant to campus accreditation purposes. For this study, we focused on accreditation-related queries, as these are the most frequently and consistently required across different departments within the university.

3.2 | Data Catalog

A data catalog is a structured metadata document that describes the tables, schemas, and columns of the target database. In this system, the data catalog serves as the primary source of domain knowledge provided to the OpenAI model during query generation. By supplying the model with a catalog of available tables and their column descriptions, we enable it to generate SQL queries that correctly reference the database schema without requiring access to the database itself. The catalog was constructed from the accreditation database schema at ITS, which contains tables related to student enrollment, academic achievement, and study program data. A sample of the catalog is shown in Table 2 , and the full entity-relationship diagram is illustrated in Figure 2 . The two primary tables used in this study are `sapto_selection_mhs_baru`, which stores information on incoming students including enrollment capacity, acceptance numbers, and re-registration statistics by study program; and `sapto_prestasi_mhs`, which records student achievement data including activity names, achievement dates, validation status, and academic year.

3.3 | OpenAI GPT-3.5 API Integration

We use the OpenAI GPT-3.5 chat completion API to generate SQL Server queries from natural language input. The chat completion architecture takes a list of messages as input, each message assigned one of three roles: `system`, `user`, or `assistant` and returns a model-generated message as output (see Figure 3). The `system` role defines a set of hard constraints on the model's

TABLE 1 Sample interview request data.

No	Original Text of the Prompted Question	Neat Text Version of the Prompted Question	Topics	Data Group	Accreditation Purposes
1	Assalamu'alaikum Bu Olyn, saya Hafidz dari DTSI ITS, ngapunten mengganggu waktunya. Saat ini DTSI sedang mengajukan akreditasi internasional ASIIN. Apakah saya boleh minta bantuan Bu Olyn untuk menarik data CPL dan CPMK mahasiswa S1 dari myITS Academics untuk semester ganjil 2022/2023 nggih Bu?	Saat ini DTSI sedang mengajukan akreditasi internasional ASIIN. Saya butuh data CPL dan CPMK mahasiswa S1 dari myITS Academics untuk semester ganjil 2022/2023.	CPMK, CPL	Mahasiswa	Ya
2	mbak Olyn ngapunten akan boleh minta raw data beban sks mengajar yang digunakan untuk perhitungan power BI kmrn?	Data beban sks mengajar yang digunakan untuk perhitungan power BI	SKS Mengajar	Dosen	Tidak
3	Non, boleh minta data tendik sarpras yang detail nggak??	Data tendik sarpras	List Nama	Tendik	Tidak
4	Mbak Olyn bisa minta tolong, saya mau minta data Tenaga Kependidikan terbaru format excel, bisa mbak olyn?	Data Tenaga Kependidikan terbaru	List Nama	Tendik	Tidak
5	Oh iya mbak kemarin Bu Sekits minta data dosen dan kegiatannya per departemen. Tujuannya untuk mengetahui siapa nama dosen yg suka mengajar, penelitian, publikasi, dll. Itu datanya minta ke Pak Radit atau bisa ke Mbak Olyn dan tim ya ?	Data dosen dan kegiatannya per departemen. Tujuannya untuk mengetahui siapa nama dosen yang suka mengajar, penelitian, publikasi, dll.	Kegiatan Dosen	Dosen	Tidak
6	Assalamu'alaikum Wr. Wb. nyuwon sewu bu, apakah memungkinkan mencari data alumni ITS atas nama Adam Priyambodo ST. Beliau alumni di ITS, cuman kami tidak tahu tahun dan jurusan apa. Beliau saat ini bekerja menjadi Senior Manajer PDAM. Besar harapan kami untuk dapat menjalin kerjasama dengan beliau, mengingat beliau alumni dari ITS juga.	Data alumni ITS atas nama Adam Priyambodo ST	Biodata	Dosen	Tidak

output. In our implementation, these constraints include generating only SQL Server syntax (not MySQL or PostgreSQL); including the schema name in every query; using `LIKE` instead of = for string comparisons; using `TOP` instead of `LIMIT` for record limiting; and always applying `DISTINCT`. These constraints were derived iteratively from observed failure modes during early testing. The `user` role provides the data catalog and the natural language query as input. The catalog is formatted as a structured

TABLE 2 Sample data catalog.

No	Schema	Table	Column	Description
1	akreditasi	mahasiswa	id_dept	Foreign key untuk join ke tabel departemen
2	akreditasi	mahasiswa	id_mapping	Foreign key untuk join ke table prodi_map
3	akreditasi	mahasiswa	jenis_kelamin	Jenis kelamin mahasiswa
4	akreditasi	mahasiswa	jurusan	Nama departemen atau jurusan mahasiswa berkuliah
5	akreditasi	mahasiswa	mahasiswa_id	Primary key dari tabel ini.
6	akreditasi	mahasiswa	nama	Nama dari mahasiswa
7	akreditasi	mahasiswa	nrp_baru	NIM atau NRP yang diketahui oleh mahasiswa
8	akreditasi	mahasiswa	nrp_lama	NIM atau NRP untuk sistem
9	akreditasi	prodi	id_departemen	Foreign key dari tabel prodi_map
10	akreditasi	prodi	id_prodi	Foreign key pada tabel ini
11	akreditasi	prodi	jenjang	Jenjang studi mahasiswa
12	akreditasi	prodi	nama	Nama dari program studi

list of available tables and their column descriptions, giving the model the context it needs to construct syntactically and semantically valid queries. The `assistant` role supplies one or more example query-answer pairs (few-shot examples), which guide the model toward the correct SQL structure for query types that are otherwise difficult to handle, such as grouped aggregation and ordering. The role prompt structure is shown in Table 3 . We selected the chat completion approach over fine-tuning because it does not require a large labeled training dataset or specialized hardware, making it practical to deploy in institutional settings with limited machine learning resources. This aligns with findings by Liu et al.[13] who demonstrated that zero-shot and few-shot prompting with GPT-3.5 can achieve competitive performance for SQL generation without any task-specific fine-tuning.

3.4 | SQL Server

Query results generated by the GPT-3.5 model were executed directly on a Microsoft SQL Server instance to verify correctness. SQL Server was chosen because it is the database management system already in use for the ITS accreditation database. Execution on the actual database served as the ground-truth validation method: a query was considered correct if it executed without errors and returned results consistent with the expected output for the given natural language request.

3.5 | Experimental Design

To evaluate the system's practical utility, we recruited eight respondents and divided them into four proficiency groups based on their daily work with SQL. Group A (two respondents) works daily as data engineers and is proficient in SQL. Group B (two respondents) works as software developers with moderate SQL experience. Group C (two respondents) works in quality assurance and uses SQL infrequently. Group D (two respondents) works as IT helpdesk staff and has no SQL knowledge. Each respondent completed two sessions: a manual session, in which they wrote SQL queries without assistance, and a chatbot session, in which they used the GPT-3.5-based chatbot. Both sessions involved the same set of 18 questions divided into three difficulty levels: simple (6 questions), medium (6 questions), and hard (6 questions), as shown in Table 4 . Questions alternated between manual and chatbot tasks at each difficulty level. Respondents were given access to the data catalog during both sessions. Performance was measured by the number of questions completed within fixed time limits (15, 30, 60, 120, 240, and 420 minutes).

FIGURE 2 Diagram table accreditation.

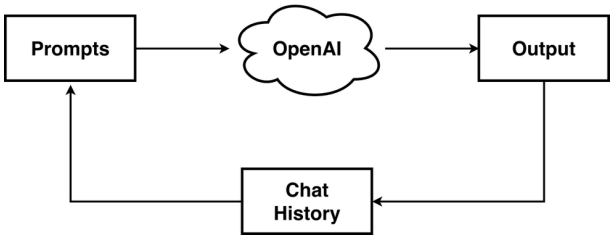

FIGURE 3 Architecture chat completion OpenAI.

4 | RESULT AND DISCUSSION

This section presents the results of the user study and discusses key findings regarding the effectiveness of the GPT-3.5-based chatbot in assisting respondents with SQL query generation.

4.1 | Task Completion Rate by Time Limit

Table 5 shows the percentage of questions each respondent was able to complete within each time limit, separately for the manual and chatbot sessions. Group A respondents, who work daily as data engineers, completed all 18 questions within 60 minutes in both sessions. The chatbot did accelerate their work at the 15-minute and 30-minute marks, but had no meaningful impact on their overall completion rate, as they were already capable of completing all tasks manually. The chatbot's benefit was most pronounced for Groups B, C, and D. In the manual session, respondents in these groups completed between 22% and 67% of questions within 60 minutes. With chatbot assistance, this rose to 67%–88% within the same time limit. Group D respondents, who had no prior SQL knowledge, required extended time to complete all manual tasks (up to 7 hours), whereas in the chatbot session they completed all tasks within 4 hours. These results indicate that chatbot assistance effectively compensates for limited SQL proficiency, enabling non-expert users to retrieve data at a speed approaching that of experienced practitioners.

TABLE 3 Sample role chat completion.

No	Role	Content
1	system	1. Buat query SQL Server untuk mengambil data di database 2. Respon yang diberikan oleh sistem hanya query tanpa penjelasan 3. Sintaks yang digunakan hanya dari SQL Server 4. Tiap query harus menyertakan skema sesuai yang didaftarkan 5. Filter menggunakan nama atau kolom string maka gunakan LIKE bukan = 6. Bahasa utama yang digunakan adalah bahasa Indonesia 7. Jika query merupakan rentang waktu maka gunakan GROUP BY 8. Jika kolom tidak tersedia di tabel maka wajib melakukan JOIN ke tabel yang memiliki kolom tersebut 9. Suatu tabel tidak dapat menggunakan kolom dari tabel yang lain 10. Gunakan DISTINCT pada tiap query 11. Dilarang menggunakan sintaks LIMIT, tetapi gunakan TOP
2	user	**Daftar tabel yang tersedia** **Skema:** akreditasi **Tabel:** gpt_mahasiswa **Deskripsi:** Tabel berisi data biodata mahasiswa. Baik mahasiswa yang masih berkuliah di ITS ataupun tidak. **Kolom:** id_dept (Foreign key untuk join ke tabel departemen), id_prodi_map (Foreign key untuk join ke tabel prodi_map), jenis_kelamin (Jenis kelamin mahasiswa), jurusan (Nama departemen atau jurusan mahasiswa berkuliah), id_mahasiswa (Primary key dari tabel ini), nama (Nama dari mahasiswa), nrp_baru (NIM atau NRP yang diketahui oleh mahasiswa.), nrp_lama (NIM atau NRP untuk sistem), id_prodi (foreign key dari tabel prodi), slta_kota (nama kota SMA/SLTA mahasiswa), tahun_angkatan (tahun mahasiswa mulai berkuliah).
3	user	Departemen dengan jumlah prestasi paling banyak di tahun :tahun

4	assistant	`SELECT TOP 1 j.nama_lengkap, COUNT(*) AS jumlah`
		`FROM akreditasi.jurusan j`
		`JOIN akreditasi.sapto_prestasi_mhs sp ON j.id = sp.id_dept`
		`WHERE sp.waktu_perolehan = :tahun`
		`GROUP BY j.nama_lengkap`
		`ORDER BY COUNT(*) DESC;`

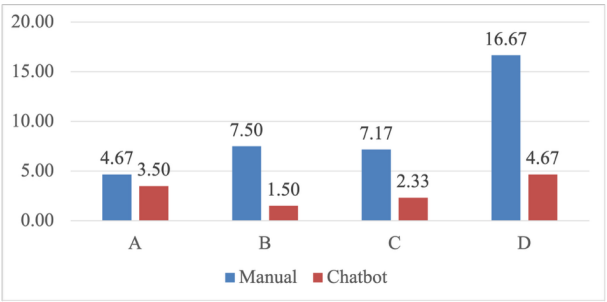

FIGURE 4 Graph of the total duration of working on simple problems.

TABLE 4 Test questions.

Code	Level	Tools	Quiz
S1	Simple	Manual	List nrp dan nama mahasiswa informatika dan sistem informasi tahun angkatan 2022.
S2	Simple	Chatbot	List nrp, nama mahasiswa laki-laki di departemen matematika tahun angkatan 2021.
S3	Simple	Manual	Data prestasi non akademik departemen sistem informasi yang diperoleh di tahun 2020 - 2022. Kolom yang ditampilkan nama kegiatan, tahun perolehan, tingkat, capaian, dan validasi.
S4	Simple	Chatbot	Data prestasi departemen teknik sipil yang diperoleh di tahun 2019-2022 dan sudah terverifikasi. Kolom yang ditampilkan nama kegiatan, tahun perolehan, tingkat, capaian, dan status validasi.
S5	Simple	Manual	Data daya tampung dan daftar ulang prodi S1 Kimia tahun 2022 Gasal.
S6	Simple	Chatbot	Data peminat dan diterima prodi S1 Fisika tahun 2021 Gasal.
M1	Medium	Manual	Jumlah mahasiswa S1 yang cuti di periode 20221 di masing-masing departemen.
M2	Medium	Chatbot	Jumlah mahasiswa S2 yang mengundurkan diri di periode 20212 di masing-masing departemen.
M3	Medium	Manual	Daftar departemen yang di tahun 2022 memiliki jumlah prestasi kurang dari 30. Data prestasi harus sudah terverifikasi.
M4	Medium	Chatbot	Daftar departemen yang di periode 20221 memiliki jumlah mahasiswa aktif kurang dari 200 orang.
M5	Medium	Manual	Departemen dengan jumlah mahasiswa aktif paling banyak di tahun 2022.
M6	Medium	Chatbot	Departemen dengan jumlah prestasi paling banyak di tahun 2022.
H1	Hard	Manual	Daftar departemen yang jumlah peminat turun dari tahun 2021 ke 2022. Kolom yang ditampilkan nama departemen, tahun, jumlah peminat.
H2	Hard	Chatbot	Daftar departemen yang jumlah daftar ulang nya turun dari tahun 2019 ke 2020. Kolom yang ditampilkan nama departemen, tahun, jumlah daftar ulang.
H3	Hard	Manual	Daftar departemen yang paling banyak mendapatkan prestasi juara 1 di masing-masing fakultas. Kolom yang ditampilkan nama fakultas, nama departemen, jumlah prestasi.
H4	Hard	Chatbot	Daftar departemen yang paling banyak mendapatkan prestasi di tingkat internasional di masing-masing fakultas. Kolom yang ditampilkan nama fakultas, nama departemen, jumlah prestasi.
H5	Hard	Manual	Daftar prestasi mahasiswa departemen sistem informasi dari tahun 2019-2022. Kolom yang perlu ditampilkan antara lain id prestasi, nama kegiatan, tahun kegiatan, tingkat (lokal, nasional, internasional), dan capaian.
H6	Hard	Chatbot	Jumlah mahasiswa berstatus "Aktif" di departemen sistem informasi dari tahun 2020-2022. Kolom yang perlu ditampilkan antara lain nama prodi, dan jumlah mahasiswa aktif. Keterangan: TS = 2022, TS-1 = 2021, TS-2 = 2020.

4.2 | Task Completion by Difficulty Level

Table 6 breaks down completion counts within 60 minutes by difficulty level (simple, medium, and hard). In the manual session, Groups B and C were able to complete all simple-level questions but struggled with medium-level tasks and failed entirely on hard-level questions. With chatbot assistance, nearly all respondents in Groups B and C completed the medium-level questions and made meaningful progress on hard-level tasks. Group D, which could only complete simple-level questions

TABLE 5 The percentage of questions that respondents can solve within the specified time.

Group	Respondent	Question Type	Working Time					
			< 15 min	< 30 min	< 60 min	< 2 hr	< 4 hr	< 7 hr
A	A1	Manual	33.33%	66.67%	100.00%	-	-	-
		Chatbot	77.78%	88.89%	100.00%	-	-	-
	A2	Manual	33.33%	66.67%	100.00%	-	-	-
		Chatbot	66.67%	88.89%	100.00%	-	-	-
B	B1	Manual	11.11%	33.33%	55.56%	-	-	-
		Chatbot	33.33%	66.67%	66.67%	-	-	-
	B2	Manual	11.11%	33.33%	44.44%	-	-	-
		Chatbot	33.33%	66.67%	66.67%	-	-	-
C	C1	Manual	11.11%	44.44%	66.67%	-	-	-
		Chatbot	66.67%	88.89%	88.89%	-	-	-
	C2	Manual	11.11%	33.33%	33.33%	-	-	-
		Chatbot	0.00%	88.89%	88.89%	-	-	-
D	D1	Manual	0.00%	0.00%	22.22%	44.44%	66.67%	100.00%
		Chatbot	22.22%	33.33%	66.67%	88.89%	100.00%	100.00%
	D2	Manual	11.11%	22.22%	33.33%	55.56%	77.78%	100.00%
		Chatbot	44.44%	66.67%	88.89%	100.00%	100.00%	100.00%

TABLE 6 Number of questions completed in less than or equal to 60 minutes.

Respondent	Manual			Chatbot		
	Simple	Medium	Hard	Simple	Medium	Hard
A1	3	3	3	3	3	3
A2	3	3	3	3	3	3
B1	3	2	0	3	3	0
B2	3	1	0	3	2	1
C1	3	3	0	3	3	2
C2	3	0	0	3	3	2
D1	2	0	0	3	3	0
D2	3	0	0	3	3	2

manually within 60 minutes, was able to complete medium-level tasks and some hard-level tasks with chatbot support. The results confirm that the chatbot's benefit scales with query complexity: it provides little additional value for simple queries (which most respondents could handle manually) but substantially increases success rates for medium and hard queries. This finding aligns with observations in the broader NL2SQL literature that GPT-based approaches perform well on moderate-complexity queries but face increasing difficulty as query structures involve nested subqueries, aggregation over multiple conditions, or PIVOT-style transformations [13].

4.3 | Effect on Query Completion Speed

Figure 4 illustrates the total time respondents spent completing simple-level questions. For Group A, chatbot assistance reduced completion time modestly, as these respondents were already fast without assistance. For Groups B, C, and D, chatbot assistance

TABLE 7 Number of problems done by reading catalog data (simple problems in manual sessions).

Respondent	Read Catalog Data?	
	No	Yes
B1	3	0
B2	1	2
C1	0	3
C2	1	2

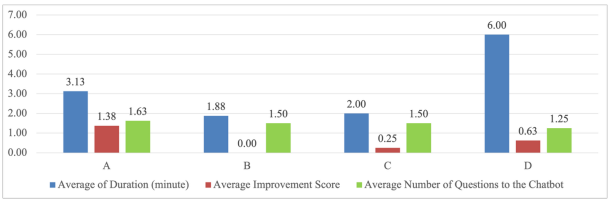

FIGURE 5 Comparison chart of respondents' interaction with the chatbot in the chatbot session (all simple questions and one medium question (M4)).

reduced completion time by a factor of 3 to 4. Notably, with chatbot assistance, Groups B, C, and D achieved completion speeds comparable to Group A's manual performance, suggesting that the system effectively narrows the competency gap between expert and non-expert users. An interesting observation concerns Groups B and C in the manual session. Group C respondents completed simple tasks faster than Group B despite having less SQL experience. Table 7 shows that this is likely explained by a behavioral difference: Group C respondents more consistently consulted the data catalog before attempting queries, whereas Group B respondents more often attempted queries from memory, leading to avoidable errors and rework.

In the chatbot session, Group A was comparatively slower than Groups B and C. This is because Group A respondents, having more SQL expertise, tended to evaluate and manually revise the chatbot's output more critically. Groups B and C, having less expertise to judge the output, more often accepted the chatbot's response without modification. As shown in Figure 5 , Group A respondents made more changes to the generated queries, while Groups B and C made fewer. Despite the higher revision rate, Group A's total completion time was still faster than Group D's, whose respondents asked the chatbot many questions without knowing how to verify or improve the output.

4.4 | Observed Failure Modes

During testing, we identified three recurring failure patterns in the GPT-3.5 model's SQL generation: **Non-deterministic outputs.** The same natural language input sometimes produced different SQL queries across respondents, and not all of these were correct. Table 8 illustrates this: for question M4, respondent B1 received an incorrect query while respondent B2 received a correct one for the same input. Similarly, for question S4, B1's result was incorrect while B2's was correct. This behavior is a known characteristic of autoregressive language models, which sample from a probability distribution rather than producing a deterministic output[13]. One mitigation strategy is to use a lower temperature setting in the API call to reduce output variability, or to apply consistency-based decoding, that is, generating multiple responses and selecting the one that appears most frequently[14]. **Incorrect SQL Server syntax.** The model consistently used the `LIMIT` keyword to restrict the number of returned rows, which is valid in MySQL and PostgreSQL but not in SQL Server, where `TOP` is the correct syntax. This was addressed in our system by adding an explicit constraint in the system prompt. However, the fact that this error persisted without the constraint suggests that GPT-3.5's training data is dominated by MySQL/PostgreSQL usage. A post-processing step that automatically substitutes `LIMIT` with `TOP` would eliminate this class of error entirely without relying on the model to follow prompt instructions. **Failure on PIVOT queries.** As shown in Table 9 , queries requiring a PIVOT transformation (e.g., question H6, which requires displaying student counts by year as separate columns) consistently produced incorrect results, even after 20 or more re-prompts.

The model was able to generate the correct aggregate calculations but failed to pivot the output into the expected columnar format. This is consistent with findings from Liu et al.[13], who noted that ChatGPT struggles with SQL constructs that require reasoning about output structure rather than just data retrieval logic. A practical mitigation is to handle PIVOT transformations as a post-processing step in application code (e.g., using Python's `pandas.pivot_table`) rather than delegating them to the SQL generation model.

TABLE 8 An example of a chatbot giving different responses to the same input.

RespondentCode		User Input	Respond Chatbot	Result
B1	M4	Tidak perlu penjelasan. Daftar departemen yang di periode 20221 memiliki jumlah mahasiswa aktif kurang dari 200 orang.	`SELECT j.nama_lengkap AS departemen, COUNT(*) AS jumlah_mahasiswa FROM akreditasi.gpt_jurusan j JOIN akreditasi.gpt_prodi_map pm ON j.id = pm.id_dept JOIN akreditasi.gpt_mahasiswa m ON pm.id = m.id_prodi_map JOIN akreditasi.gpt_status_mahasiswa sm ON m.nrp_lama = sm.nrp_lama WHERE sm.status = "Aktif" AND sm.periode_semester = "Gasal" AND sm.tahun_akademik = "2021/2022" GROUP BY j.nama_lengkap HAVING COUNT(*) < 200`	Salah
B2	M4	Tidak perlu penjelasan. Daftar departemen yang di periode 20221 memiliki jumlah mahasiswa aktif kurang dari 200 orang.	`SELECT j.nama_lengkap AS departemen, COUNT(m.id_mahasiswa) AS jumlah_mahasiswa FROM akreditasi.gpt_jurusan j JOIN akreditasi.gpt_mahasiswa m ON j.id = m.id_dept JOIN akreditasi.gpt_status_mahasiswa sm ON m.nrp_lama = sm.nrp_lama WHERE sm.status = "Aktif" AND sm.periode_semester = "20221" GROUP BY j.nama_lengkap HAVING COUNT(m.id_mahasiswa) < 200;`	Benar
B1	S4	Tidak perlu penjelasan. Data prestasi departemen teknik sipil yang diperoleh di tahun 2019-2022 dan sudah terverifikasi. Kolom yang ditampilkan nama kegiatan, tahun perolehan, tingkat, capaian, dan status validasi.	`SELECT nama_kegiatan, waktu_perolehan, tingkat, capaian, status_validasi FROM akreditasi.gpt_sapto_prestasi_mhs spm JOIN akreditasi.gpt_jurusan j ON spm.id_dept = j.id WHERE j.nama_lengkap = "Departemen Teknik Sipil" AND waktu_perolehan BETWEEN 2019 AND 2022 AND status_validasi IN ("Valid oleh sistem", "Terverifikasi");`	Salah

Continued on next page

TABLE 8 An example of a chatbot giving different responses to the same input. *(Continued)*

RespondentCode		User Input	Respond Chatbot	Result
B2	S4	Tidak perlu penjelasan. Data prestasi departemen teknik sipil yang diperoleh di tahun 2019-2022 dan sudah terverifikasi. Kolom yang ditampilkan nama kegiatan, tahun perolehan, tingkat, capaian, dan status validasi.	`SELECT nama_kegiatan, waktu_perolehan, tingkat, capaian, status_validasi FROM akreditasi.gpt_sapto_prestasi_mhs WHERE id_dept IN ( SELECT id FROM akreditasi.gpt_jurusan WHERE nama_lengkap = "Departemen Teknik Sipil" ) AND waktu_perolehan BETWEEN 2019 AND 2022 AND status_validasi = "Terverifikasi";`	Benar
C1	S6	Tidak perlu penjelasan. Data peminat dan diterima prodi S1 Fisika tahun 2021 Gasal.	`SELECT peminat, diterima FROM akreditasi.gpt_sapto_seleksi_mhs_baru WHERE tahun = 2021 AND periode = "Gasal" AND id_prodi_map IN ( SELECT id FROM akreditasi.gpt_prodi_map WHERE nama LIKE "%S1 Fisika%" );`	Benar
C2	S6	Tidak perlu penjelasan. Data peminat dan diterima prodi S1 Fisika tahun 2021 Gasal.	`SELECT peminat, diterima FROM akreditasi.gpt_sapto_seleksi_mhs_baru WHERE id_prodi_map = ( SELECT id FROM akreditasi.gpt_prodi_map WHERE nama = "S1 Fisika") AND tahun = 2021 AND periode_semester = "Gasal";`	Salah

TABLE 9 The number of questions given by respondents to the chatbot and the score of improvements made by respondents to the chatbot query in question code H6.

Group	Respondent	Number of Inquiries to Chatbot	Improvement Score
A	A1	2	3
	A2	21	5
D	D1	22	3
	D2	5	5

5 | CONCLUSION

This study presented the implementation and evaluation of a chatbot system for generating SQL queries from natural language inputs, built on the OpenAI GPT-3.5 chat completion API. The system was applied to a real-world institutional context at Institut Teknologi Sepuluh Nopember (ITS), Surabaya, targeting an accreditation database that staff members query regularly in response to informal, loosely worded data requests. The results of a human-subject evaluation involving eight respondents across four SQL proficiency groups demonstrate that chatbot assistance substantially improves query completion rates and reduces completion time for users with limited SQL experience. Respondents in Groups B, C, and D completed 67%–88% of all tasks within 60 minutes with chatbot assistance, compared to 22%–67% in the manual session. For Group D, who had no

prior SQL knowledge, the chatbot enabled completion of medium- and hard-level queries that were entirely out of reach in the manual session. For expert users in Group A, the chatbot offered speed advantages at shorter time limits but did not change overall completion rates, as these respondents were already capable of completing all tasks manually. Three recurring failure modes were identified: non-deterministic outputs for identical inputs, incorrect SQL Server syntax (specifically, the use of `LIMIT` instead of `TOP`), and inability to generate correct PIVOT queries. These limitations point to concrete directions for future improvement. The `LIMIT/TOP` issue can be fully resolved through post-processing rules or explicit prompt constraints. Non-deterministic outputs can be mitigated by lowering the model's temperature parameter or applying consistency-based decoding strategies. PIVOT queries are best handled outside the SQL generation model, either through application-layer post-processing or by decomposing the query into simpler steps. Future work could extend this system in several directions. First, the prompt design and data catalog construction approach demonstrated here could be adapted to other domain-specific databases within the same institution or across other universities. Second, evaluating the system with more recent models (e.g., GPT-4 or open-source alternatives) would clarify how much of the observed limitation is specific to GPT-3.5 versus inherent to the prompting-based NL2SQL paradigm. Third, a larger-scale user study with a more diverse respondent pool would strengthen the generalizability of the findings beyond the eight participants in this study.

ACKNOWLEDGMENT

The authors would like to thank the Directorate of Research and Community Service of Institut Teknologi Sepuluh Nopember (ITS) for funding and supporting this research under grant No. 1645/PKS/ITS/2023. The authors also extend their gratitude to the Directorate of Technology and Information System Development (DPTSI) of ITS for providing access to the institutional database and supporting the data collection process.

CREDIT

Radityo Prasetianto Wibowo and Rahmad Santosa contributed to conceptualization, methodology, writing of the original draft, and supervision. Hartantya Ainiyatus Tsaniyah contributed to formal analysis and investigation. Adetiya Bagus Nusantara and Yoga Ari Tofan contributed to writing, review, and editing. Resources were provided by Rahmad Santosa, Hartantya Ainiyatus Tsaniyah, Yoga Ari Tofan, and Adetiya Bagus Nusantara.

References

1. Long H, Cao D. BERT-based Text-to-SQL Generation Method With Question-table Content Enhancement and Template Filling. In: 2021 2nd International Conference on Information Science and Education (ICISE-IE); 2021. p. 969–973.

2. Simitsis A, Koutrika G, Ioannidis Y. Pr'ecis: From Unstructured Keywords as Queries to Structured Databases as Answers. VLDB Journal 2008 Jan;17(1):117–149.

3. Li F, Jagadish HV. Constructing an Interactive Natural Language Interface for Relational Databases. Proceedings of the VLDB Endowment 2014 Sep;8(1):73–84.

4. Baik C, Jagadish HV, Li Y. Bridging the Semantic Gap with SQL Query Logs in Natural Language Interfaces to Databases. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE); 2019. p. 374–385.

5. Saha D, Floratou A, Sankaranarayanan K, Minhas UF, Mittal AR, "Ozcan F. ATHENA: An Ontology-driven System for Natural Language Querying Over Relational Data Stores. Proceedings of the VLDB Endowment 2016 Aug;9(12):1209–1220.

6. Fan Y, Ren T, He Z, Wang XS, Zhang Y, Li X. GenSql: A Generative Natural Language Interface to Database Systems. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE); 2023. p. 3603–3606.

7. Popescu AM, Etzioni O, Kautz H. Towards a Theory of Natural Language Interfaces to Databases. In: Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03) New York, NY, USA: Association for Computing

Machinery; 2003. p. 149–157.

8. Singh G, Solanki A. An Algorithm to Transform Natural Language Into SQL Queries for Relational Databases. Selforganizology 2016 sep;https://www.semanticscholar.org/paper/An-algorithm-to-transform-natural-language-into-SQL-Singh-Solanki/a50cd64211689b4fca6adee3031e35ba42c31765, accessed: Jun. 24, 2026.

9. Xu X, Liu C, Song D. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv preprint arXiv:171104436 2017 Nov;.

10. Yu T, Zhang R, Yasunaga M, Tan YC, Lin XV, Li S, et al. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. In: Riloff E, Chiang D, Hockenmaier J, Tsujii J, editors. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing Brussels, Belgium: Association for Computational Linguistics; 2018. p. 1653–1663.

11. Katsogiannis-Meimarakis G, Koutrika G. A Survey on Deep Learning Approaches for Text-to-SQL. VLDB Journal 2023 Jul;32(4):905–936.

12. Qin B, et al. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. arXiv preprint arXiv:220813629 2022 Aug;.

13. Liu A, Hu X, Wen L, Yu PS. A Comprehensive Evaluation of ChatGPT's Zero-shot Text-to-SQL Capability. arXiv preprint arXiv:230313547 2023 Mar;.

14. Nan L, et al. Enhancing Few-shot Text-to-SQL Capabilities of Large Language Models: A Study on Prompt Design Strategies. arXiv preprint arXiv:230512586 2023 May;.

15. Liu X, et al. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? arXiv preprint arXiv:240805109 2024 Aug;.

16. Kurniawan A, Abdiansah A, Utami AS. NL2SQL for Chatbot with Semantic Parsing Using Rule-Based Methods. Sriwijaya Journal of Informatics and Applications 2024;5(1).

17. Kanburoğlu AB, Tek FB. TUR2SQL: A Cross-Domain Turkish Dataset For Text-to-SQL. In: 2023 8th International Conference on Computer Science and Engineering (UBMK); 2023. p. 206–211.

18. Ionescu VM, Enescu MC. Using ChatGPT for Generating and Evaluating Online Tests. In: 2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI); 2023. p. 1–6.

How to cite this article: Rahmad S., Hartantya A. T., Yoga A. T, Adetiya B. N., Radityo P. W., (2026), Implementation of Chatbot for Generating Natural Language to SQL Queries, *IPTEK The Journal for Technology and Science*, *37(2):100-113*.