

يسرى محمد علي الفران

Yusra salam final report .docx

 Midocean University

Document Details

Submission ID

trn:oid::2945:406442860

Submission Date

Aug 8, 2026, 6:00 PM GMT+3

Download Date

Aug 8, 2026, 6:38 PM GMT+3

File Name

Yusra salam final report .docx

File Size

6.9 MB

72 Pages

15,419 Words

101,500 Characters

83% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely revised by AI.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.



117 AI-generated 83%

Likely generated or likely generated and revised by AI.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely human generated as AI generated and likely AI generated as human generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.



MIDOCEAN UNIVERSITY
COLLEGE OF INFORMATICS
MASTER OF SCIENCE IN ARTIFICIAL INTELLIGENCE

**Design and Implementation of an Intelligent AI Chatbot
with an
Integrated Product Recommendation System for E-
Commerce Platforms**

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Artificial Intelligence

Prepared by

Researcher	Student ID
Yusra Elfaran	2021202562
Salam Khouli	261000006

Supervised by

Dr. Khaled Eskaf

College of Informatics, Midocean University
Moroni, Comoros

2026

Approval Page

This thesis, entitled “Design and Implementation of an Intelligent AI Chatbot with an Integrated Product Recommendation System for E-Commerce Platforms,” was prepared by Yusra Elfaran and Salam Khouli in partial fulfillment of the requirements for the Master of Science in Artificial Intelligence at the College of Informatics, Midocean University.

Supervisor: Dr. Khaled Eskaf

Examiner 1: _____

Examiner 2: _____

Date: _____

Declaration of Originality

We declare that this thesis presents the work undertaken for the stated master's project and that all external ideas, methods, and publications used in the manuscript are acknowledged through in-text citation and the reference list. The experimental results, screenshots, software artifacts, and model files reported in the thesis are drawn from the supplied project archive and the executed verification activities. This thesis has not been submitted in the same form for another degree.

Yusra Elfaran: _____

Salam Khouli: _____

Acknowledgements

The researchers express their sincere appreciation to Dr. Khaled Eskaf for his academic supervision, constructive observations, and emphasis on methodological transparency throughout the project. Appreciation is also extended to the College of Informatics at Midocean University for providing the academic framework in which the study was completed. The researchers are grateful to their families and colleagues for their encouragement during the implementation, experimentation, and documentation stages.

الملخص

الأهداف: تصميم وتنفيذ وتقييم روبوت محادثة ذكي للتجارة الإلكترونية يدمج تصنيف النوايا بالبحث عن المنتجات والانحدار اللوجستي، وتوثيق TF-IDF بخط أساس يعتمد على DistilBERT والتوصية القائمة على المحتوى، مع مقارنة كتجربة خفيفة منفصلة. المنهجية: اتبعت الدراسة تصميمًا تجريبيًا تطبيقيًا. نُظمت بيانات النوايا من BERT-Tiny عبارة إلى 11,586 عبارة موزعة على 20 فئة، وقُسمت طبقًا إلى 8,110 للتدريب، و1,738 للتحقق، 14,500 NVIDIA Tesla T4، باستخدام وحدة معالجة رسومات Google Colab على DistilBERT و1,738 للاختبار. دُرّب وربط بكتالوج SQLite وFastAPI وReact وقورن بخط الأساس على مجموعة الاختبار نفسها. نُفذ التطبيق باستخدام يضم 8,000 منتج. وشمل التحقق النهائي 12 سيناريو محادثة مختارًا، ومجموعة انحدار وظيفية من 52 حالة، و15 فحصًا نهائيًا للتشغيل والواجهات البرمجية والأصول المحلية وسلامة البيانات والأخطاء المنطقية. النتائج: حقق BERT-مرجحة 0.9564، مقابل 0.8993 و0.8978 لخط الأساس. وحقق F1 دقة 0.9563 ودرجة DistilBERT دقة 0.8548 في تجربة منفصلة لا تمثل مقارنة معمارية مضبوطة. نجحت حالات الانحدار الـ52 والفحوص Tiny التشغيلية الـ15 دون إخفاق بعد تصحيح عيوب التنفيذ، بما فيها الاعتماد الخارجي لصورة الهيرو، تهيئة قاعدة البيانات، التحقق من المدخلات، واستقلال تشغيل المودلين محليًا. اقتصر تقييم التوصية على مؤشرات تشخيصية بديلة دون أحكام بشرية أو سجلات تفاعل حقيقية؛ لذا لا تثبت النتائج التخصيص أو رضا المستخدم. التوصيات: توثيق مصدر البيانات، وتكرار التدريب ببذور متعددة، وجمع محادثات حقيقية بموافقة المستخدمين، وتحسين بيانات المنتجات والأسعار، وإجراء اختبارات الأمن والحمل والاستعادة قبل النشر.

الكلمات الدالة: روبوت محادثة للتجارة الإلكترونية، تصنيف النوايا، DistilBERT، التوصية القائمة على المحتوى، التجارة الحوارية.

Abstract

Objectives: To design, implement, and evaluate an intelligent e-commerce chatbot that integrates intent classification with product search and content-based recommendation; compare DistilBERT with a TF-IDF logistic-regression baseline; and document BERT-Tiny as a separate lightweight experiment. **Methodology:** An applied experimental design was used. Cleaning reduced 14,500 intent utterances to 11,586 examples across 20 classes, stratified into 8,110 training, 1,738 validation, and 1,738 test examples. DistilBERT was trained on Google Colab with an NVIDIA Tesla T4 GPU and compared with the baseline on the same test set. The application used React, FastAPI, SQLite, and an 8,000-product catalog. Final verification comprised 12 selected conversational scenarios, a 52-case functional regression suite, and 15 focused runtime, API-validation, recommendation, and hidden-issue checks. **Results:** DistilBERT achieved 0.9563 accuracy and 0.9564 weighted F1, versus 0.8993 accuracy and 0.8978 weighted F1 for the baseline. BERT-Tiny achieved 0.8548 accuracy in a separate experiment that was not treated as a controlled architectural comparison. All 52 regression cases and all 15 focused runtime, API, recommendation, and hidden-issue checks passed after correcting implementation defects, including the external hero dependency, database initialization order, input validation, and offline model loading. **Recommendation evaluation used proxy diagnostics without human relevance judgments or real interaction logs; therefore, the findings do not establish personalization or user satisfaction. Recommendations:** Document dataset provenance, repeat training with multiple seeds, collect consented real conversations, improve product and price completeness, and conduct security, load, and recovery testing before production deployment.

Keywords: e-commerce chatbot, intent classification, DistilBERT, content-based recommendation, conversational commerce.

Table of Contents

Approval Page	i
Declaration of Originality	ii
Acknowledgements	iii
Abstract	v
Table of Contents	vi
List of Figures	x
List of Tables.....	xi
List of Abbreviations.....	xii
1.1 Background of the Study	1
1.2 Research Problem	3
1.3 Aim and Objectives	4
1.4 Research Questions.....	4
1.5 Significance of the Study.....	5
1.6 Scope, Boundaries, and Limitations	6
1.7 Research Contributions.....	7
1.8 Thesis Organization	8
1.9 Chapter Summary	8
2.1 Introduction	9
2.2 Theoretical Foundations	9
2.2.1 Transformer Self-Attention.....	11
2.2.2 TF-IDF and Logistic Regression	11
2.2.3 Content-Based Recommendation.....	12
2.3 E-Commerce Chatbots and Intent Classification.....	12
2.4 Conversational Recommendation.....	13
2.5 Hybrid Statistical and Rule-Based Systems	13
2.6 Previous Empirical Studies.....	14

2.7 Comparative Analysis and Research Gap	14
2.8 Chapter Summary	16
3.1 Research Design	17
3.2 System Requirements	17
3.3 Data Resources and Preprocessing	18
3.3.1 Primary Intent Dataset	19
3.3.2 Auxiliary BERT-Tiny Dataset Pipeline	19
3.3.3 Product Catalog.....	20
3.4 Model Training	20
3.4.1 Controlled Baseline.....	20
3.4.2 DistilBERT Fine-Tuning	20
3.4.3 BERT-Tiny Auxiliary Experiment.....	20
3.5 Product Retrieval and Recommendation	21
3.6 Proposed System Architecture.....	22
3.6.1 UML Sequence Model.....	23
3.6.2 Use-Case Model.....	24
3.6.3 Data-Flow Model	25
3.7 Software Implementation	26
3.7.1 Frontend	26
3.7.2 Backend and API	26
3.7.3 Database.....	27
3.7.4 Chat Routing and Business Logic.....	27
3.8 Evaluation Protocol	27
3.9 Reproducibility, Ethics, and Privacy	28
3.10 Chapter Summary	28
4.1 Dataset and Training Results	29
4.2 Controlled Classification Results	31
4.3 Per-Class Analysis	32

4.4 Auxiliary BERT-Tiny Results	34
4.5 Recommendation Diagnostics	34
4.6 Chatbot and API Experiments	36
4.6.1 Executed Chatbot Scenarios.....	36
4.6.2 Automated API Verification	38
4.6.3 Interface Verification	38
4.6.4 Final Runtime Verification and Hidden-Issue Corrections	39
4.7 Discussion of Research Questions.....	44
4.8 Limitations and Threats to Validity.....	46
4.9 Chapter Summary	47
5.1 Conclusion	48
5.2 Summary of Contributions	49
5.3 Recommendations	50
5.3.1 Data	50
5.3.2 Modeling	50
5.3.3 Recommendation	50
5.3.4 Deployment.....	50
5.3.5 Multilingual Extension.....	50
5.4 Future Work Roadmap	51
5.5 Final Statement	51
References	52
Appendix A: Intent Labels and Operational Meanings.....	54
Appendix B: Executed Chatbot Scenario Matrix	55
Appendix C: Final Runtime Validation	56
Appendix D: Installation and Execution Guide	57
D.1 Backend	57
D.2 Frontend.....	57
D.3 Verification	57

Appendix E: Screenshot Evidence Manifest.....	58
Appendix F: Moodle Submission Checklist.....	59

List of Figures

Figure (1): Proposed architecture of the implemented e-commerce chatbot.	23
Figure (2): UML sequence diagram for chatbot request processing.	24
Figure (3): UML use-case diagram for the implemented e-commerce chatbot.	25
Figure (4): Level-1 data-flow diagram for the implemented e-commerce chatbot.	26
Figure (5): Distribution of the 20 intent classes after primary data cleaning.	31
Figure (6): DistilBERT training and validation behavior across four epochs.	31
Figure (7): Controlled comparison between TF-IDF logistic regression and DistilBERT.	32
Figure (8): DistilBERT per-class F1-scores across the twenty intents.	33
Figure (9): Normalized DistilBERT confusion matrix on the primary test set.	34
Figure (10): Product catalog price and rating distributions.	36
Figure (11): Recommendation retrieval diagnostics for six representative queries.	36
Figure (12): Executed chatbot tests for product, typo, policy, and out-of-scope requests.	39
Figure (13): Running home, catalog-search, and product interfaces.	40
Figure (14): Research dashboard and evaluation interface.	41
Figure (15): Final homepage using the approved beauty-store identity and packaged local hero asset.	43
Figure (16): Executed DistilBERT price-constrained recommendation with budget-compliant, product-type-relevant results.	44
Figure (17): Executed product-comparison scenario with distinct visible titles and category-consistent images.	44
Figure (18): Executed typographical-error recovery showing hybrid normalization and final routing behavior.	45
Figure (19): Controlled negative-budget validation message with no product retrieval.	45
Figure (20): Controlled out-of-catalog response that avoids returning unrelated beauty products.	46

List of Tables

Table (1): Research questions and corresponding evidence	5
Table (2): Comparison of selected related studies	14
Table (3): Functional requirements	19
Table (4): Non-functional requirements	19
Table (5): Data resources and analytical roles	20
Table (6): Primary intent-data cleaning and split	20
Table (7): Model configurations	22
Table (8): System components and responsibilities	23
Table (9): Backend API endpoints	27
Table (10): Evaluation measures	28
Table (11): Controlled classification performance on the same 1,738-item test set	32
Table (12): Selected class-level DistilBERT results	34
Table (13): Auxiliary BERT-Tiny results	35
Table (14): Recommendation diagnostics at $k = 5$	36
Table (15): Executed DistilBERT chatbot scenarios, part A	37
Table (16): Executed DistilBERT chatbot scenarios, part B	38
Table (17): Automated application verification	40
Table (18): Final implementation defects, corrections, and verification evidence	41
Table (19): Research-question findings	46
Table (A.1): Intent labels and operational meanings	56
Table (B.1): Executed chatbot scenarios	57
Table (C.1): Final runtime, API, asset, and hidden-issue validation results	58
Table (E.1): Screenshot evidence files	61

List of Abbreviations

Abbreviations used in the thesis

Abbreviation	Meaning
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CPU	Central Processing Unit
DFD	Data Flow Diagram
F1	Harmonic Mean of Precision and Recall
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
NLP	Natural Language Processing
NLU	Natural Language Understanding
REST	Representational State Transfer
RQ	Research Question
SQL	Structured Query Language
SQLite	Embedded Relational Database Engine
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
UML	Unified Modeling Language

CHAPTER 1

INTRODUCTION

Background, research problem, objectives, questions, significance, and study boundaries

1.1 Background of the Study

Conversational interfaces are increasingly used to reduce the effort required to search, compare, and purchase products online. A conventional catalog requires users to translate an information need into category navigation, filters, and repeated queries. A conversational interface can instead accept a natural-language request such as “recommend waterproof mascara under 20 dollars,” infer the requested action, extract product constraints, and return a structured response. The practical value of such a system depends on more than generating fluent text: the assistant must identify the user’s intent, connect the intent to reliable application functions, retrieve suitable items, and preserve a transparent boundary between real functionality and simulated store policies.

Intent classification is therefore a central component of task-oriented chatbots. Transformer architectures introduced contextual representations that can model an utterance as a sequence rather than as an unordered collection of terms (Vaswani et al., 2017). BERT extended this idea through bidirectional pre-training and demonstrated the effectiveness of fine-tuning for classification (Devlin et al., 2019). DistilBERT reduced the size and inference cost of BERT while retaining much of its language-understanding capability, making it relevant to applications that require a balance between accuracy and deployment efficiency (Sanh et al., 2019). Nevertheless, a high test score alone does not establish that a chatbot is useful. The classifier must be integrated with routing logic, product retrieval, database operations, and user-interface feedback.

Conversational recommendation introduces an additional challenge. Surveys of conversational recommender systems emphasize that dialogue understanding, preference elicitation, recommendation strategy, and evaluation are interdependent rather than isolated tasks (Gao et al., 2021; Jannach et al., 2021). The present project addresses a bounded version of this problem. It uses an intent classifier to determine the required shopping function and a content-based retrieval component to rank products from a local catalog. The implementation includes a React/Vite frontend, a FastAPI backend, a SQLite database, two deployable BERT-

family classifiers, deterministic routing safeguards, and product recommendation based on TF-IDF similarity with metadata reranking.

Previous applied chatbot research has shown that domain-specific virtual assistants require an explicit knowledge structure, an interaction interface, and a response-routing mechanism that connects user requests to appropriate system functions (Mohamed et al., 2020). Consistent with this principle, the present system combines learned intent classification with deterministic routing and application services rather than treating the chatbot as an isolated text classifier. This distinction is practically important because high offline classification accuracy does not necessarily guarantee correct end-to-end behavior: errors may arise during entity extraction, routing, database retrieval, response construction, or interface rendering. Conversely, a successful final response may result from deterministic refinement of an incorrect raw prediction. Classifier performance and complete application behavior are therefore evaluated as separate forms of evidence.

The study contributes an end-to-end, reproducible prototype and evaluates it at three levels. First, it reports a controlled comparison between TF-IDF logistic regression and DistilBERT on the same cleaned intent dataset and the same held-out test split. Second, it reports BERT-Tiny separately because that model was trained through an auxiliary pipeline with different cleaning and sampling decisions. Third, it verifies the behavior of the running application through API checks and executed chat scenarios. This separation is important because model evaluation, recommendation diagnostics, and software functional testing answer different questions and must not be combined into a single accuracy claim.

The implementation perspective adopted in this thesis treats the chatbot as a socio-technical software artifact rather than a single predictive model. The classification component provides a probabilistic interpretation of an utterance, but business operations remain constrained by explicit application logic, available data, and the user interface. This approach is important in commerce because an apparently minor error can change a search request into a cart action, confuse a policy question with an order command, or return products that violate a stated budget. The system therefore records the raw classifier output, the final route selected by the application, the confidence value, any override reason, and the resulting response type.

The selected domain is e-commerce product discovery and store assistance. The catalog is predominantly composed of beauty products and contains titles, text descriptions, ratings, review counts, partially available prices, image references, and a stored recommendation score. The chatbot supports twenty intent categories covering product discovery, product

information, recommendation, comparison, price constraints, cart actions, order services, payment, delivery, customer support, store information, feedback, and out-of-scope requests. This bounded scope makes controlled evaluation possible while still representing a realistic mixture of informational and transactional language.

1.2 Research Problem

Many e-commerce chatbot prototypes are evaluated either as text classifiers without a functioning store or as interface demonstrations without verifiable language-understanding results. This separation creates a methodological gap. A classifier may achieve high offline accuracy but fail to trigger the correct business function when faced with misspellings, price expressions, follow-up requests, or out-of-domain messages. Conversely, a visually complete interface may rely primarily on hard-coded keywords while presenting model confidence as if it were the sole source of the decision. The research problem is therefore how to design and evaluate a transparent system in which a learned intent model, deterministic safety rules, product retrieval, and application services cooperate without confusing their respective contributions.

The project also contains two trained transformer models produced by different experimental pipelines. Directly ranking them as if they were trained on identical data would be scientifically misleading. The primary DistilBERT experiment cleaned the raw corpus by removing noisy and duplicate samples and used 11,586 utterances with a 70/15/15 split. The auxiliary BERT-Tiny pipeline retained sanitized samples, produced 12,851 records, and capped training examples per intent. The study must therefore identify which results are directly comparable and which are descriptive only. A second problem concerns recommendation evaluation: the catalog has product text and metadata but no ground-truth user-item interactions, so retrieval proxy metrics cannot be interpreted as user preference accuracy.

The central research problem can therefore be stated as follows: how can a BERT-family intent classifier be integrated with product retrieval, deterministic routing, and database-backed e-commerce services so that both the machine-learning evidence and the final application behavior remain transparent, verifiable, and academically defensible?

1.3 Aim and Objectives

The main objective of this study is to design, implement, and evaluate an intelligent e-commerce chatbot that integrates intent classification with content-based product retrieval and database-backed application functions. The specific objectives are:

1. To document the primary e-commerce intent dataset, its cleaning procedure, class distribution, and stratified training, validation, and test split.
2. To fine-tune DistilBERT for multiclass e-commerce intent classification and compare its performance with a TF-IDF logistic-regression baseline using the same processed dataset and held-out test set.
3. To evaluate overall and class-level classification performance using accuracy, weighted and macro precision, recall, F1-score, learning curves, and the confusion matrix.
4. To report the BERT-Tiny model as a separate lightweight auxiliary experiment without presenting its result as a controlled architectural comparison with DistilBERT.
5. To implement an integrated application that maps predicted intents to product search, content-based recommendation, product information, price filtering, policy, cart, order, and customer-support functions.
6. To evaluate the running chatbot through executed user scenarios and automated API tests while distinguishing raw intent predictions from final application routes.
7. To identify the data, recommendation, experimental, privacy, and deployment limitations that affect the interpretation and generalization of the study results.

These objectives are addressed in the Results and Discussion section through dataset and training analysis, controlled model comparison, class-level error analysis, the separate BERT-Tiny report, recommendation diagnostics, executed chatbot scenarios, API verification, and critical analysis of the study limitations.

1.4 Research Questions

RQ1. How accurately does DistilBERT classify the 20 predefined e-commerce intents compared with a TF-IDF logistic-regression baseline evaluated on the same processed dataset and held-out test split?

RQ2. Which intent classes remain the most difficult to classify, and what error patterns are revealed by the class-level metrics and confusion matrix?

RQ3. How does the deployed chatbot translate a raw model prediction into a final application route for product search, recommendation, product information, policy, cart, order, and customer-support functions?

RQ4. What do the executed chatbot scenarios and API tests reveal about raw intent predictions, deterministic routing refinement, response behavior, functional completion, and recorded latency?

RQ5. What data, recommendation, experimental, and deployment limitations prevent the current results from being interpreted as evidence of personalized recommendation quality or production-level effectiveness?

Table (1): Research questions and corresponding evidence

Research Question	Primary Evidence	Thesis Location
RQ1	Controlled DistilBERT vs. TF-IDF experiment	Table (11), Figures (6)-(7)
RQ2	Per-class report and confusion analysis	Table (12), Figures (8)-(9)
RQ3	Architecture, routing, and implementation trace	Figures (1)-(4), Tables (8)-(9)
RQ4	Executed chat scenarios and automated API tests	Tables (15)-(18), Figures (12)-(20)
RQ5	Recommendation diagnostics and methodological limitations	Table (14), Figures (10)-(11), Section 4.8

1.5 Significance of the Study

Scientifically, the study demonstrates why experimental boundaries matter in applied artificial-intelligence research. It distinguishes a controlled model comparison from an auxiliary experiment, separates offline classification metrics from functional scenario completion, and labels recommendation measures as retrieval proxies rather than preference accuracy. These distinctions reduce the risk of overstating results and make the findings reproducible from the artifacts contained in the project archive. The study also provides class-level evidence instead of relying only on overall accuracy, which is particularly relevant because the intent distribution is not perfectly balanced and some categories are semantically close.

Practically, the study provides a complete local prototype that can be executed without relying on a paid external language-model API. The architecture connects natural-language requests to real product records, store policies, simulated cart and order operations, and visible product cards. This makes the system useful as a reference implementation for academic projects and small-scale experiments in conversational commerce. The local design also supports inspection of the model files, code, database, and test results, allowing reviewers to verify what the system actually performs.

The work is also important from a user-experience perspective. A customer request may combine an action, a product type, and a constraint in one sentence. Correctly handling such a request reduces interaction steps and can make product discovery more accessible. At the same time, the results show that robust behavior cannot be attributed to the classifier alone. Typo normalization and deterministic refinement were necessary in selected cases, demonstrating that dependable applied systems often require an auditable hybrid of statistical prediction and explicit business logic.

At the scientific level, the study demonstrates a disciplined way to separate evidence that is often merged in applied artificial-intelligence projects. Offline accuracy measures the classifier on a held-out set. Recommendation diagnostics describe whether the retrieval engine returns and ranks candidates. Functional tests determine whether selected messages activate the intended software path. A user study would measure perceived usefulness, trust, and satisfaction, but such evidence is not available in the current project. Maintaining these distinctions reduces overclaiming and supports reproducibility.

At the engineering level, the study provides a local, inspectable implementation based on widely used open technologies. The architecture can be executed without a paid generative-model API, and its main behavior can be traced through source code, database records, model configuration files, test reports, and screenshots. This is particularly relevant to academic projects that require verifiable ownership of the implementation and the ability to demonstrate the system during examination.

1.6 Scope, Boundaries, and Limitations

The thematic boundary is task-oriented conversational commerce: product discovery, recommendation, comparison, cart and order guidance, delivery, payment, account support, feedback, and out-of-scope detection. The study does not evaluate open-domain conversation or generative answers.

The linguistic boundary is English-language intent classification. Arabic appears only in the Arabic abstract and keywords; the deployed classifier was not trained or evaluated on Arabic or code-mixed requests.

The data boundary comprises the intent files, trained models, 8,000-product SQLite catalog, notebooks, metrics, screenshots, and test reports contained in the supplied project archive. The original provenance and annotation protocol of the intent corpus were not documented in the archive, and this is treated as a limitation rather than inferred.

The experimental boundary is a fixed-seed comparison between TF-IDF logistic regression and DistilBERT on one stratified split. BERT-Tiny is reported from a different cleaning and sampling pipeline and therefore is descriptive, not a controlled architectural comparison.

The software boundary is a research prototype executed in a local environment. Functional tests establish that endpoints and selected scenarios worked at test time; they do not certify production security, scalability, privacy, accessibility, transaction integrity, or long-term reliability.

The prototype was evaluated using project datasets and researcher-defined test messages rather than private customer conversations. No claim is made that the system is ready to collect or retain personal information in production. A future deployment should apply data minimization, explicit consent, secure storage, access control, retention limits, and anonymization for user messages, confidence scores, interaction logs, cart records, and order-related data. User interactions should not be reused for model training without an approved consent and governance procedure. These safeguards are especially important because deterministic routes and confidence values can reveal behavioral patterns even when the underlying message does not contain an obvious identifier.

The catalog boundary is also significant. Of the 8,000 product records, only 2,534 contain a usable positive price, leaving 5,466 records (68.3%) without price data. Price-filtering experiments therefore operate on a restricted subset and cannot be interpreted as full-catalog budget coverage. The catalog is also highly concentrated in the All Beauty category, with only a small Premium Beauty subset, which limits generalization to broad multi-category e-commerce platforms.

1.7 Research Contributions

- A controlled comparison between TF-IDF logistic regression and DistilBERT on the same cleaned dataset and the same held-out test split.
- A transparent distinction between the primary DistilBERT experiment and the separate BERT-Tiny pipeline.
- An implemented full-stack e-commerce chatbot that connects intent prediction with search, recommendation, policy, cart, order, and support functions.
- An auditable distinction between raw model intent and final application route, including deterministic override reasons.
- A three-layer evaluation protocol comprising model metrics, recommendation diagnostics, and executed software scenarios.

- A documented analysis of limitations involving dataset provenance, catalog completeness, lack of human relevance judgments, and absence of production testing.

1.8 Thesis Organization

Chapter 1 presents the study background, problem, objectives, research questions, significance, scope, and contributions. Chapter 2 reviews the theoretical foundations and related work on Transformers, intent classification, conversational recommendation, out-of-scope detection, and hybrid intelligent systems. Chapter 3 explains the applied research design, datasets, preprocessing, model training, product retrieval method, proposed architecture, software implementation, and evaluation protocol. Chapter 4 reports and discusses the experimental results, recommendation diagnostics, chatbot scenarios, API verification, and answers to the research questions. Chapter 5 summarizes the conclusions, contributions, recommendations, and future work. The appendices provide the intent taxonomy, detailed test scenarios, API verification summary, and execution guidance.

1.9 Chapter Summary

This chapter established the motivation for a transparent e-commerce chatbot that integrates machine-learning predictions with explicit application logic. It defined the methodological problem, specified the research objectives and questions, and clarified the boundaries that govern interpretation of the results. These foundations guide the review of related work in the next chapter.

CHAPTER 2

RELATED WORK

Theoretical foundations, empirical studies, and the research gap

2.1 Introduction

This chapter reviews the concepts and previous studies that support the implemented system. The review is organized around four connected themes: contextual language representation, e-commerce intent classification, conversational recommendation, and hybrid routing through explicit rules and structured knowledge. The purpose is not to reproduce textbook descriptions but to identify the design decisions that are relevant to the current artifact and the limitations that remain unresolved in prior research.

2.2 Theoretical Foundations

The Transformer architecture replaced recurrent processing with self-attention, allowing each token to weight other tokens in the sequence and enabling parallel training (Vaswani et al., 2017). BERT applies bidirectional Transformer encoding and is pre-trained through masked-language modeling and sentence-level objectives before task-specific fine-tuning (Devlin et al., 2019). In intent classification, a classification head is trained over the contextual representation of an utterance. The benefit over bag-of-words methods is that word meaning can depend on surrounding terms and ordering; however, performance still depends on label quality, domain coverage, and similarity among intent definitions.

DistilBERT is a compressed BERT-family model produced through knowledge distillation. The original study reported a 40% size reduction and faster inference while retaining most of BERT's language-understanding performance (Sanh et al., 2019). These properties make

DistilBERT a reasonable candidate for a research prototype that runs inference on a local CPU after GPU-based training. BERT-Tiny represents a more aggressive reduction in model capacity. In the current project it serves as an optional low-resource model, but its experimental result cannot isolate architecture effects because its preprocessing, sample size, and optimization settings differ from the primary pipeline.

A traditional TF-IDF model represents documents using term importance based on within-document frequency and corpus-level rarity (Salton & Buckley, 1988). Combined with logistic regression, it provides an interpretable and computationally inexpensive baseline. Such a baseline is important because a transformer should be justified by measurable gains on the same test data rather than by model novelty alone. In this study, the controlled baseline uses unigram and bigram TF-IDF features, class-balanced logistic regression, and the exact primary test split used for DistilBERT.

Intent classification is operationally defined as assigning one of 20 predefined labels to a user utterance. The final route is the application action selected after optional deterministic refinement. These are not always identical: the raw intent is the model prediction, whereas the final route reflects the hybrid system decision. Functional case completion means that the application returned the expected type of behavior for a selected test prompt; it is not an estimate of model accuracy over an independently sampled population.

Aly, Eskaf, and Selim (2017) also demonstrate the value of intelligent expert-system design in decision-support applications, where rule-based reasoning and structured knowledge translate user inputs into actionable recommendations. This supports the present study's use of transparent routing safeguards alongside statistical intent prediction.

The recommendation component is content-based because the project archive does not contain real user-item interaction histories. Content-based methods rank items by similarity between query terms and product descriptions and can incorporate item metadata such as rating and popularity (Adomavicius & Tuzhilin, 2005). The Amazon Reviews 2023 resource provides review and item metadata at large scale and is the documented source family for the product catalog used by the project (Hou et al., 2024). In the implemented system, TF-IDF similarity identifies lexical relevance, while rating, review count, and a stored recommendation score contribute to reranking. This design is reproducible but cannot learn personalized preferences without user behavior data.

For this study, chatbot implementation means the complete path from a user message to a visible application result. The path includes normalization, classifier inference, deterministic

refinement, business-function routing, database or recommendation access, response construction, and frontend rendering. Reliability is assessed through held-out model evaluation and deterministic functional tests. Production reliability would additionally require user studies, repeated performance trials, load testing, security assessment, monitoring, and evaluation on data collected after deployment; these are outside the current study boundaries.

Out-of-scope detection is particularly important in a commerce assistant because a closed intent inventory cannot safely absorb every request. Larson et al. (2019) demonstrated that classifiers that perform well on in-scope categories can still struggle with unsupported inputs. For the present system, out-of-scope is therefore both a learned label and an application boundary: it prevents the chatbot from implying competence beyond shopping, order, delivery, payment, and support functions.

Dialogue-system evaluation literature also cautions against relying on a single automatic score. Deriu et al. (2021) distinguish component-level measurements from end-to-end and human evaluation. This distinction motivates the study's three-layer protocol: offline intent classification, recommendation diagnostics, and executed software scenarios. Each layer is reported separately because it measures a different object.

2.2.1 Transformer Self-Attention

The core Transformer operation calculates attention weights between token representations. In simplified form, the attention output is obtained by projecting the input into query, key, and value matrices and applying scaled dot-product attention. The mechanism allows each token to incorporate information from other positions in the utterance, which is particularly useful when intent depends on the relationship between product terms, actions, and constraints.

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V \quad (1)$$

2.2.2 TF-IDF and Logistic Regression

TF-IDF remains a strong baseline because it converts documents into sparse lexical vectors and can be trained efficiently. For a term t in document d , the weight increases with term frequency and decreases when the term occurs in many documents. Logistic regression then estimates the probability of each intent label from the resulting feature vector. This baseline is interpretable and establishes whether the additional complexity of a Transformer produces a measurable gain.

$$TF-IDF(t, d) = TF(t, d) \times \log(N / DF(t)) \quad (2)$$

2.2.3 Content-Based Recommendation

Content-based retrieval represents both the user request and product text in a shared vector space. The implemented engine uses cosine similarity to measure lexical alignment, then combines similarity with title matching, full-text matching, and a stored recommendation score. This method can operate without user histories, but it cannot infer long-term personal preferences. It is also vulnerable to lexical mismatch when a query and a relevant item use different words.

$$\cosine(q, p) = (q \cdot p) / (||q|| ||p||) \quad (3)$$

$$combined\ score = 0.50\ similarity + 0.30 \quad (4)$$

$$title\ score + 0.10\ text\ score + 0.10\ stored\ recommendation\ score$$

2.3 E-Commerce Chatbots and Intent Classification

Task-oriented e-commerce chatbots differ from open-domain systems because the supported functions are explicitly defined. The assistant must identify whether a message requests a search, recommendation, comparison, price constraint, cart action, order service, policy explanation, or support response. Intent taxonomies make evaluation possible, but they also introduce boundary problems: semantically similar categories may overlap, and unsupported requests may be forced into the nearest known class unless out-of-scope handling is designed deliberately.

Sharma (2020) showed that inter-utterance context can improve intent classification in an e-commerce voice assistant. The result is relevant because follow-up messages such as “show me cheaper ones” cannot be interpreted correctly without the preceding product request. The current prototype stores limited structured context, including recent products and selected items, but its classifier is trained on single utterances. Contextual classification therefore remains an important extension.

Kulkarni et al. (2023) examined label-efficient conversational intent classification and emphasized the difficulty of grammatical errors and mixed-language requests. The present implementation includes deterministic typo normalization and evaluates selected misspelled inputs, but the training corpus remains English-only and the project does not exploit unlabeled

production conversations. The study therefore contributes implementation evidence rather than a solution to label scarcity or multilingual robustness.

Larson et al. (2019) demonstrated that out-of-scope detection is a separate challenge from in-scope classification. Even high-performing classifiers can assign unsupported messages to known intents with high confidence. The current system includes an out_of_scope label and a deterministic boundary for clearly unrelated prompts, but broader real-world evaluation would require diverse unsupported requests and calibrated rejection thresholds.

2.4 Conversational Recommendation

Gao et al. (2021) organized conversational recommender-system research around preference elicitation, dialogue understanding, strategy, exploration, and evaluation. Jannach et al. (2021) similarly emphasized that a conversational recommender combines multiple components rather than a single ranking algorithm. The current system implements a bounded form of conversational recommendation: users can express product types and constraints in natural language, while the backend retrieves and reranks products. It does not learn a user profile, optimize a multi-turn dialogue policy, or adapt from interaction outcomes.

Hou et al. (2024) introduced Amazon Reviews 2023 and language-item representations for retrieval and recommendation. The project catalog belongs to this source family and demonstrates how product titles, descriptions, ratings, and review counts can support local retrieval. However, the implemented catalog is a small beauty-focused subset, and the exact transformation from the public source into the SQLite database is not fully documented in the supplied archive. This provenance limitation is reported rather than reconstructed from assumptions.

Evaluation is especially important. Coverage and average similarity can confirm that the engine returns candidates, but they do not establish that users consider those candidates relevant, diverse, useful, or satisfactory. Human relevance judgments and interaction outcomes would be required for stronger claims. The present study therefore reports recommendation results as proxy diagnostics.

2.5 Hybrid Statistical and Rule-Based Systems

The integration of statistical prediction and explicit decision logic is consistent with expert-system research. Aly et al. (2017) demonstrated how structured knowledge and fuzzy reasoning can convert user information into actionable advice. Mohamed et al. (2020) showed that a domain-specific chatbot requires a knowledge structure, an interaction interface, and a

routing mechanism. These principles support the current architecture, where the model proposes an intent and deterministic safeguards inspect explicit expressions such as prices, checkout terms, shipping terms, and product keywords before selecting the final route.

A hybrid design has two advantages. First, it can protect critical business functions from errors that are easy to detect through explicit patterns. Second, it provides an audit trail explaining why the final route differed from the raw prediction. The same design also creates a risk: if only the final route is reported, the rules can conceal model errors. The thesis therefore records both values and treats the override rate as an operational indicator rather than an accuracy measure.

2.6 Previous Empirical Studies

Table (2): Comparison of selected related studies

Study	Focus	Method/Data	Key Finding	Gap Relative to This Study
Devlin et al. (2019)	BERT pre-training and fine-tuning	Large pretrained Transformer evaluated across NLP tasks	Bidirectional contextual representation improves downstream performance	Not domain-specific and does not address application integration
Sanh et al. (2019)	DistilBERT compression	Knowledge distillation from BERT	Reduced size and faster inference with retained capability	Does not evaluate e-commerce routing or recommendation
Larson et al. (2019)	Intent and out-of-scope evaluation	Large intent dataset with unsupported requests	Out-of-scope detection remains difficult	No e-commerce application or business-function integration
Sharma (2020)	E-commerce voice-assistant intent classification	Context-enhanced classifier	Inter-utterance context improves follow-up interpretation	No product recommendation or full-stack verification
Mohamed et al. (2020)	Virtual assistant for academic advising	Knowledge structure, interface, and routing	Demonstrates applied assistant integration	Different domain and no Transformer comparison
Gao et al. (2021)	Conversational recommender systems	Survey of methods and challenges	Evaluation must include dialogue and recommendation dimensions	Survey; does not provide a directly comparable artifact
Jannach et al. (2021)	Conversational recommendation	Systematic survey	Recommendation, dialogue management, and interaction design are interdependent	No single implementation benchmark
Kulkarni et al. (2023)	Label-efficient intent classification	Semi-supervised conversational classification	Addresses limited labels and noisy user language	No integrated store or recommendation engine
Hou et al. (2024)	Language-item retrieval and Amazon Reviews 2023	Large product/review resource and retrieval models	Shows rich language-item representations	Current project uses a smaller classical local subset

2.7 Comparative Analysis and Research Gap

Sharma (2020) studied intent classification in an English e-commerce voice assistant and showed that inter-utterance context can improve the interpretation of follow-up requests. The study is directly relevant because shopping conversations often contain incomplete second

turns such as a filter request that depends on a previous product search. The current prototype preserves limited structured context, including recent product identifiers and selected items, but its classifier itself is single-utterance. This difference identifies a clear extension: future versions should evaluate contextual intent models rather than relying mainly on routing state.

Kulkarni et al. (2023) addressed label-efficient conversational intent classification and emphasized challenges arising from grammatical errors and mixed-language user queries. Their findings support the need to test more than clean held-out utterances. The present project includes typo normalization and executed misspelled prompts, but its training and formal test corpus are English-only. Unlike semi-supervised approaches, it does not exploit unlabeled interaction logs. Thus, the current work contributes implementation evidence while leaving label efficiency and multilingual robustness unresolved.

Gao et al. (2021) organized conversational recommender-system challenges around preference elicitation, multi-turn strategy, dialogue understanding, exploration, and evaluation. Jannach et al. (2021) likewise showed that conversational recommendation combines recommendation, dialogue management, and interaction design. These surveys highlight a limitation of evaluating recommendation only through similarity. The implemented prototype can respond conversationally and return ranked products, but it does not learn long-term preferences or optimize dialogue strategy from interaction outcomes. Its metrics must therefore be described as retrieval diagnostics.

Hou et al. (2024) introduced Amazon Reviews 2023 and language-item representations for retrieval and recommendation. That work illustrates the scale and richness possible when product metadata and review language are linked. The present project uses a small, beauty-focused subset of item metadata and a classical TF-IDF ranker rather than a learned semantic retriever. This makes local execution simpler, but the catalog's missing prices and lexical ambiguities limit the quality of price filtering and product-type matching.

The research gap addressed here is not a new transformer architecture. It is the rigorous integration and evaluation of a bounded e-commerce chatbot artifact. Existing studies often optimize either intent models or conversational recommendation methods. The present study connects an empirically evaluated classifier to a functioning application, reports the contribution of deterministic routing, and explicitly separates controlled classification evidence, auxiliary model results, recommendation proxies, and software verification. This framing provides a clearer account of what the implemented system has and has not demonstrated.

Larson et al. (2019) provided a large intent dataset that explicitly included out-of-scope queries and showed that rejecting unsupported requests is harder than classifying supported intents. That result is relevant to the present out_of_scope category and to the observation that it was the lowest-F1 class in the DistilBERT test report. However, the current study differs by connecting the classifier to product, order, and policy actions in a working store prototype.

Deriu et al. (2021) surveyed dialogue-system evaluation and emphasized that automatic metrics do not fully capture user experience or task success. Similarly, Jannach et al. (2021) and Gao et al. (2021) noted that conversational recommendation should be evaluated across dialogue quality, recommendation usefulness, and interaction outcomes. These findings justify the paper's refusal to interpret retrieval proxies or selected scenario completion as user satisfaction.

The resulting research gap is an auditable, end-to-end e-commerce prototype in which model accuracy, rule-based routing, product retrieval, and application verification are reported with clear experimental boundaries. The contribution is not a new Transformer architecture. It is a transparent implementation and evaluation framework showing how a trained intent model operates inside a full application and where the evidence remains insufficient for production claims.

The present study extends this applied direction by combining chatbot-routing principles reflected in Mohamed et al. (2020) with structured decision-support logic consistent with Aly et al. (2017), while applying both to an e-commerce chatbot and content-based product-retrieval context.

The research gap is therefore an auditable end-to-end evaluation framework rather than a new neural architecture. The thesis contributes a controlled DistilBERT experiment, an auxiliary lightweight model report, a transparent routing layer, a content-based retrieval engine, and executed application tests within one coherent artifact. The next chapter explains how these components were designed and implemented.

2.8 Chapter Summary

This chapter established the theoretical basis for contextual intent classification, lightweight Transformer models, content-based retrieval, and hybrid routing. The reviewed literature demonstrates both the value of the selected methods and the need to separate classifier metrics, recommendation evidence, and end-to-end behavior. These conclusions directly inform the methodology and proposed framework presented in Chapter 3.

CHAPTER 3

METHODOLOGY AND PROPOSED FRAMEWORK

Actual implementation, model training, data processing, software architecture, and experiments

3.1 Research Design

The study adopted an applied experimental software-engineering design. The unit of analysis for the primary machine-learning experiment was an English e-commerce utterance paired with one of twenty intent labels. The unit of analysis for product retrieval was a catalog record. The unit of analysis for functional verification was an executed user message or automated API assertion. This design permits evaluation at model, retrieval, and application levels while maintaining clear boundaries between them.

The project archive, trained model folders, notebooks, SQLite database, application source code, configuration files, metrics, and screenshots were treated as the primary research artifacts. Results were accepted only when they could be traced to these artifacts or reproduced through execution. Where provenance or experimental detail was not documented, the thesis identifies the limitation instead of inferring unsupported information.

3.2 System Requirements

Table (3): Functional requirements

ID	Requirement
----	-------------

ID	Requirement
FR-01	Accept a natural-language shopping or support message through the web interface.
FR-02	Normalize common spelling variations and extract explicit constraints such as price values.
FR-03	Predict one of twenty supported e-commerce intents and return a confidence value.
FR-04	Apply deterministic routing safeguards and record any override reason.
FR-05	Search the product catalog by textual content and product terms.
FR-06	Generate content-based product recommendations and rerank candidates.
FR-07	Filter products by a user-specified maximum price when price data are available.
FR-08	Return product details and compare selected products.
FR-09	Support simulated cart, checkout, order, payment, delivery, and support functions.
FR-10	Detect and respond to requests outside the supported store domain.
FR-11	Expose evaluation metrics and model information through backend endpoints.
FR-12	Render structured JSON responses as text, product cards, actions, and updated context.

Table (4): Non-functional requirements

ID	Quality Attribute	Requirement
NFR-01	Traceability	Record raw intent, final route, confidence, and override reason.
NFR-02	Local execution	Core intent and retrieval functions operate without a paid external generative-model API.
NFR-03	Maintainability	Separate frontend, API, model service, recommendation service, and database layers.
NFR-04	Reproducibility	Retain model configurations, metrics, test reports, and executable source code.
NFR-05	Performance	Return interactive responses within the local test environment; no production SLA is claimed.
NFR-06	Privacy	Use research data and researcher-defined messages; no private customer dataset is used.
NFR-07	Security boundary	Simulated payment and order actions must not be presented as production transactions.
NFR-08	Usability	Provide product cards, clear feedback, and accessible routes to search, compare, and act.

3.3 Data Resources and Preprocessing

Table (5): Data resources and analytical roles

Resource	Size	Role in the study	Qualification
Primary intent corpus	14,500 raw; 11,586 cleaned	Controlled baseline and DistilBERT experiment	Original source and annotation process are not documented in the archive

Resource	Size	Role in the study	Qualification
Auxiliary intent pipeline	12,851 cleaned; 3,939 capped training cases	BERT-Tiny deployment experiment	Different cleaning, sampling, and split; descriptive only
Product catalog	8,000 product records	Search, filtering, and content-based recommendation	5,466 records have no usable price
Functional chat set	12 prompts tested with two models	Raw prediction, final routing, response, and latency inspection	Purposefully selected cases; not a random accuracy sample
Automated application suite	15 assertions	Endpoint and business-scenario verification	Does not cover security, load, usability, or recovery

3.3.1 Primary Intent Dataset

The primary notebook identified four fields: id, text, label, and set, with twenty raw labels. The cleaning procedure normalized whitespace, removed empty records, retained utterances between 3 and 300 characters, removed rows matching a documented profanity/noise pattern, removed duplicate text-label pairs, and retained labels with at least twenty samples. This process reduced the primary corpus from 14,500 to 11,586 utterances, removing 2,914 rows. The cleaned dataset was encoded and split using stratification with a fixed random seed of 42 into 8,110 training, 1,738 validation, and 1,738 test samples.

Table (6): Primary intent-data cleaning and split

Stage	Rows	Description
Raw intent data	14,500	20 intent labels
Cleaned data	11,586	79.9% retained; 2,914 records removed
Training set	8,110	70.0% of cleaned data
Validation set	1,738	15.0% of cleaned data
Test set	1,738	15.0% of cleaned data

3.3.2 Auxiliary BERT-Tiny Dataset Pipeline

A second notebook produced a separate cleaned dataset of 12,851 utterances and a 70/15/15 split of 8,995 training, 1,928 validation, and 1,928 test records. Training was capped at 200 examples per intent, resulting in 3,939 training examples used by BERT-Tiny. Because this pipeline differs from the primary dataset in cleaning, sampling, and split, its result is reported as auxiliary deployment evidence rather than a controlled comparison.

3.3.3 Product Catalog

The SQLite catalog contains 8,000 product records: 7,905 in All Beauty and 95 in Premium Beauty. Average rating is 4.19, the median rating is 4.3, and the average review count is 486.46. Only 2,534 products contain a usable positive price; 5,466 products have missing price values. The product catalog supports search, filtering, and content-based recommendation, but the missing-price rate limits the candidate pool for budget requests.

3.4 Model Training

3.4.1 Controlled Baseline

The baseline uses unigram and bigram TF-IDF features with a maximum vocabulary of 50,000, minimum document frequency of two, and class-balanced logistic regression. The model is trained on the same 8,110 primary training examples and evaluated on the same 1,738 held-out test examples used for DistilBERT. This design makes the comparison controlled with respect to the data split.

3.4.2 DistilBERT Fine-Tuning

The primary model is distilbert-base-uncased. Utterances are tokenized to a maximum sequence length of 96. Training uses four epochs, a training batch size of 16, an evaluation batch size of 32, and a learning rate of 2×10^{-5} . The supplied notebook records training on Google Colab using an NVIDIA Tesla T4 GPU. Validation loss and classification metrics are monitored after each epoch, and the final model folder contains configuration, tokenizer, label maps, and safetensors weights.

3.4.3 BERT-Tiny Auxiliary Experiment

The auxiliary model uses prajjwal1/bert-tiny with a maximum sequence length of 64, ten epochs, batch size 16, learning rate 5×10^{-4} , and a cap of 200 training examples per intent. The archive identifies this model as deployable and computationally lightweight. Its test results are not used to rank the architecture against DistilBERT because the underlying data pipeline is different.

Table (7): Model configurations

Approach	Main configuration	Training/Test	Interpretation
TF-IDF + logistic regression	TF-IDF 1–2 grams; max 50,000; min_df=2; class-balanced; seed 42	8,110 / 1,738	Controlled baseline
DistilBERT	distilbert-base-uncased; length 96; 4 epochs; batch 16/32;	8,110 / 1,738	Controlled primary model

Approach	Main configuration	Training/Test	Interpretation
	LR 2×10^{-5}		
BERT-Tiny	prajjwal1/bert-tiny; length 64; 10 epochs; batch 16; LR 5×10^{-4}	3,939 / 1,928	Separate auxiliary pipeline

3.5 Product Retrieval and Recommendation

At application startup, the recommendation service loads the SQLite product table and fits a TF-IDF vectorizer to the stored product_text field with up to 8,000 features. The user query is normalized, common typographical variants are replaced, and product terms are expanded through a small synonym dictionary. Candidate products are selected using title and product-text matching. Cosine similarity is then calculated between the query vector and candidate vectors.

The combined score implemented in the source code assigns 50% weight to TF-IDF similarity, 30% to title-term coverage, 10% to product-text term coverage, and 10% to a stored recommendation score. Final sorting also considers average rating and review count. Budget requests exclude products without a positive price and apply the extracted maximum price before ranking. This method is transparent and reproducible, but it is not personalized and remains sensitive to missing prices, duplicates, and lexical mismatch.

$$score = 0.50s + 0.30t + 0.10x + 0.10r \quad (5)$$

where s is cosine similarity, t is title-term coverage, x is product-text term coverage, and r is the stored recommendation score. Rating and review count are used as secondary sorting criteria rather than as direct terms in the combined score.

3.6 Proposed System Architecture

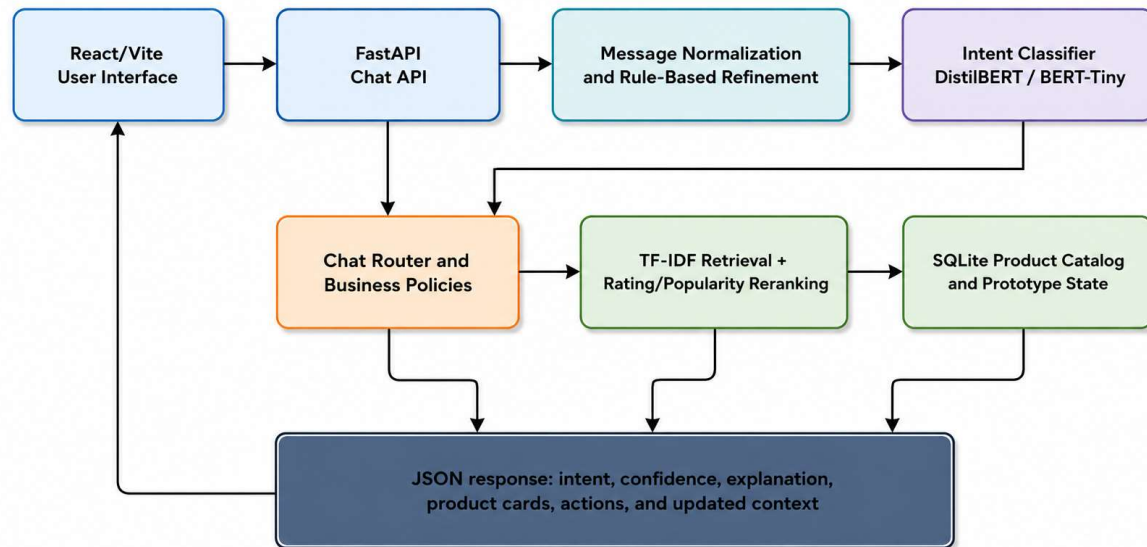


Figure (1): Proposed architecture of the implemented e-commerce chatbot.

Table (8): System components and responsibilities

Component	Responsibility
React/Vite UI	Collects messages, renders responses, product cards, model metadata, and user actions.
FastAPI Chat API	Receives requests, validates payloads, coordinates model inference, routing, retrieval, and response construction.
Normalization and Refinement	Normalizes misspellings, detects price and product expressions, and applies documented deterministic safeguards.
Intent Classifier	Produces the raw intent label and confidence using DistilBERT or BERT-Tiny.
Chat Router and Business Policies	Maps the final route to search, recommendation, product, cart, order, policy, support, or fallback behavior.
Recommendation Engine	Performs TF-IDF retrieval, lexical filtering, scoring, rating/popularity reranking, and price filtering.
SQLite Database	Stores product records and prototype tables for users, cart items, orders, chat logs, and recommendation logs.
Response Formatter	Returns structured JSON containing intent, confidence, explanation, product cards, actions, and updated context.

3.6.1 UML Sequence Model

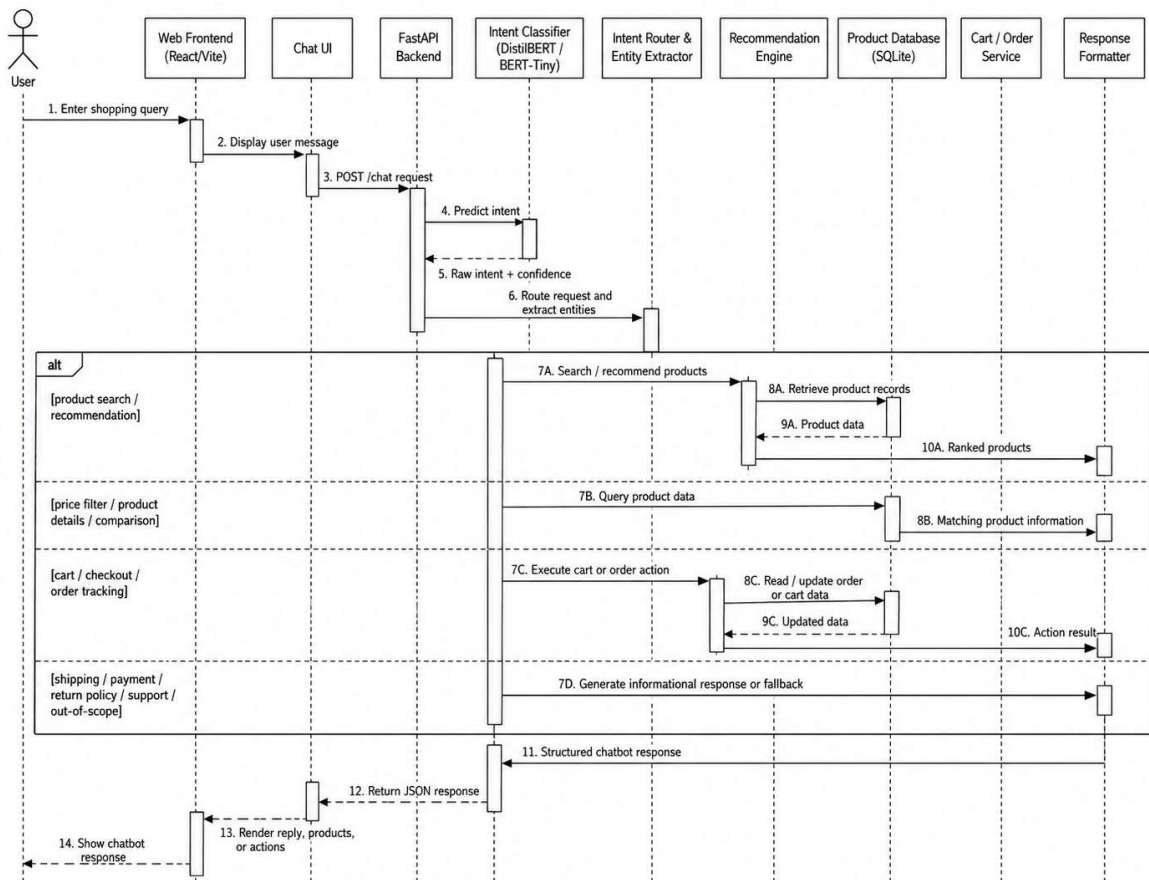


Figure (2): UML sequence diagram for chatbot request processing.

The sequence model shows that a user message passes through the web frontend and chat interface to the FastAPI backend. The classifier returns a raw intent and confidence, after which the router extracts entities and selects one of several application paths. Product-related requests access retrieval and database services, while cart and order requests access stateful services. Informational and out-of-scope requests use controlled response generation. The structured response is returned to the frontend for rendering.

3.6.2 Use-Case Model

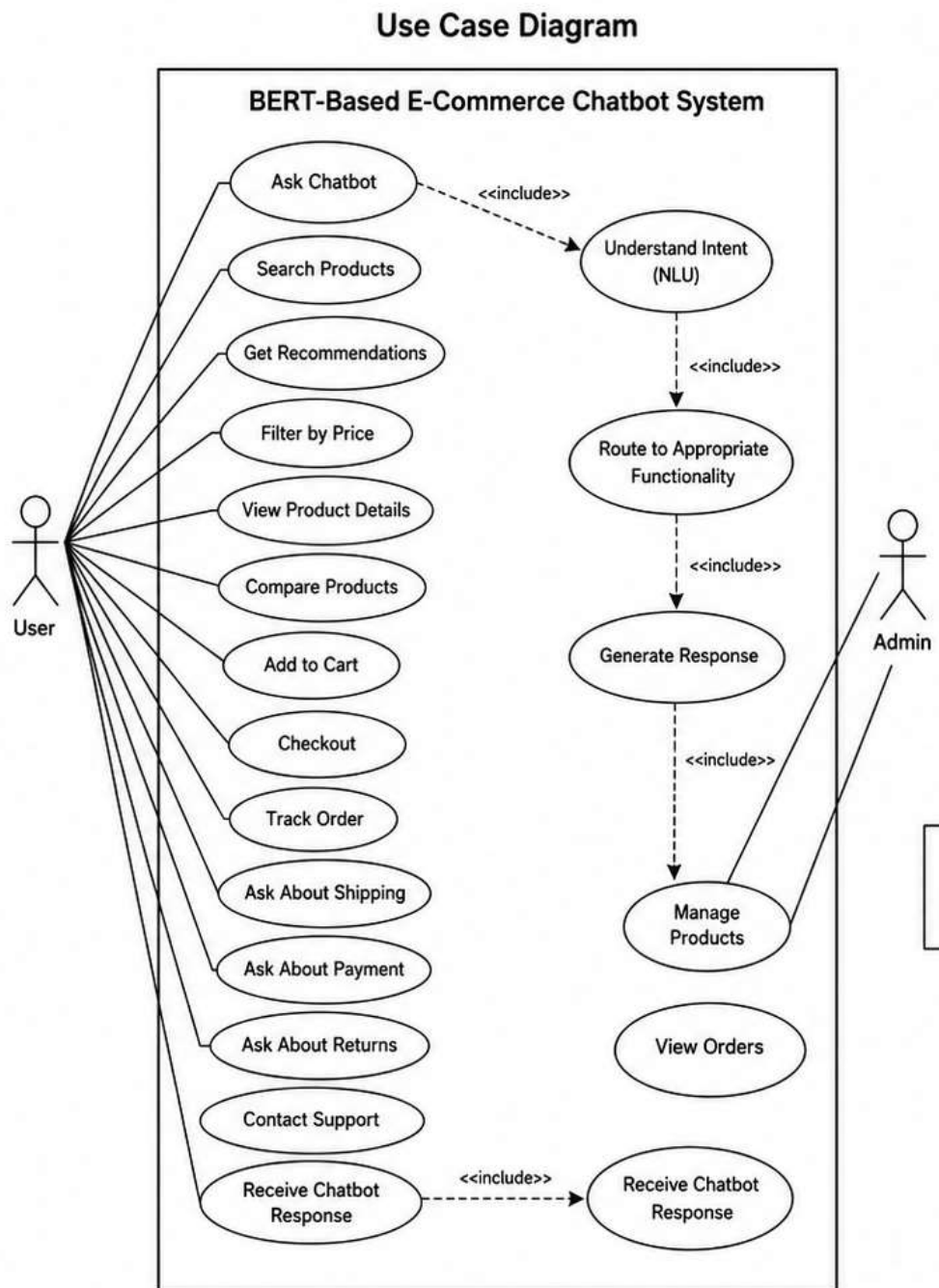


Figure (3): UML use-case diagram for the implemented e-commerce chatbot.

3.6.3 Data-Flow Model

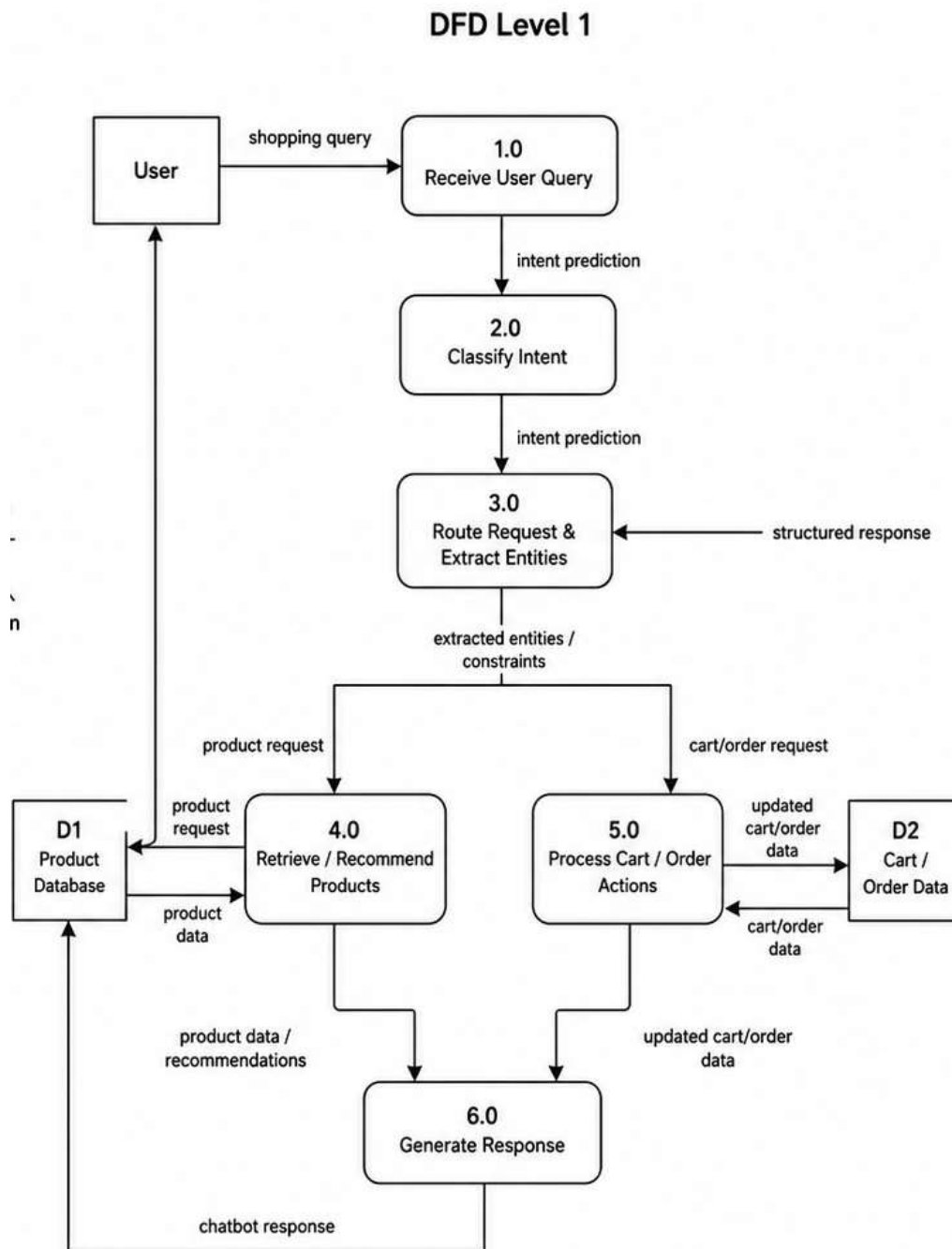


Figure (4): Level-1 data-flow diagram for the implemented e-commerce chatbot.

3.7 Software Implementation

3.7.1 Frontend

The frontend is implemented with React 19 and Vite 6. It provides the home page, product catalog, chat interface, model-selection controls, cart-related interactions, login demonstration, and research dashboard. Product cards display locally assigned category images, product titles, ratings, prices when available, and actions for details, comparison, and cart operations. The interface communicates with the backend through JSON requests and renders the structured response returned by the chat endpoint.

3.7.2 Backend and API

The backend is implemented with FastAPI and initializes the SQLite database at application startup. Cross-origin access is limited to the local Vite development addresses. Routers separate product access, recommendations, chat operations, and evaluation metrics. The chat route normalizes input, calls the selected model, applies routing rules, invokes the relevant service, and returns a structured response. This separation improves inspectability and allows the individual endpoints to be tested independently.

Table (9): Backend API endpoints

Endpoint	Purpose
GET /api/health	Service availability check.
GET /api/products	Paginated catalog retrieval and filtering.
GET /api/products/categories	Product-category retrieval.
GET /api/products/{parent asin}	Single-product details.
POST /api/recommendations	Recommendation request through the retrieval service.
POST /api/chat	End-to-end chatbot request.
POST /api/chat/intent	Intent-only inference.
GET /api/chat/models	Available deployed classifiers.
GET /api/evaluation/intent	Primary intent metrics.
GET /api/evaluation/recommendation	Recommendation diagnostic metrics.
GET /api/evaluation/dataset-cleaning	Dataset-cleaning summary.

3.7.3 Database

The embedded SQLite database contains products, users, cart_items, orders, chat_logs, and recommendation_logs tables. The products table is the active evidence source for search and retrieval. The remaining tables support prototype state and simulated application behavior. At the time of inspection, the product table contained 8,000 records, while the user, cart, and order tables were empty before interaction. This confirms that the project is a research prototype rather than a populated commercial store.

3.7.4 Chat Routing and Business Logic

The routing layer preserves both the raw intent and final route. Explicit conditions detect price expressions, product keywords, details requests, checkout terms, shipping and payment vocabulary, tracking references, and common typographical errors. The design allows deterministic refinement when the model output conflicts with an unambiguous operational expression. Override reasons are returned for auditability. This behavior is evaluated separately from classifier accuracy.

3.8 Evaluation Protocol

Table (10): Evaluation measures

Measure	Interpretation
Accuracy	Overall proportion of correctly classified test examples.
Weighted Precision/Recall/F1	Class metrics weighted by class support; useful for the observed distribution.
Macro Precision/Recall/F1	Unweighted mean across classes; highlights performance on smaller classes.
Confusion Matrix	Distribution of predicted labels for each true label.
Coverage@5	Whether five items were returned for each selected retrieval query; a system diagnostic.
Mean Similarity@5	Average TF-IDF similarity of the top five candidates; a lexical diagnostic.
Mean Reranked Score@5	Average combined retrieval score for the top five candidates.
Functional Completion	Whether the selected scenario returned the expected response type after the full pipeline.
Override Rate	Proportion of scenarios in which the final route differs from the raw intent.
Recorded Latency	Observed response time in the local sequential test environment; not a hardware-independent benchmark.

The controlled classifier experiment compares TF-IDF logistic regression and DistilBERT using identical held-out examples. The BERT-Tiny result is reported separately. Recommendation evaluation uses six representative queries at $k = 5$ because no human relevance labels or user-item interactions are available. Application verification consists of

twelve selected chatbot scenarios and fifteen automated assertions. These layers answer different research questions and are not combined into a single performance value.

3.9 Reproducibility, Ethics, and Privacy

The archive contains source code, model configurations, trained weights, notebooks, metrics, screenshots, and test reports. The primary configuration records the DistilBERT model name, maximum sequence length, performance values, and test-set size. The BERT-Tiny configuration records its model name, data paths, epochs, batch size, sequence length, learning rate, and sampling cap. The Python requirements file lists dependency families but does not pin exact versions; this is a reproducibility limitation. The frontend package file does record major package versions.

The functional evaluation used researcher-defined messages and project data rather than private customer conversations. A production system would require consent, data minimization, retention controls, access controls, secure logging, and a clear policy governing whether conversations may be reused for training. Payment, delivery, and order responses in the current system are simulations and must not be interpreted as real transactions.

3.10 Chapter Summary

This chapter documented the actual implementation and experimental method. It described the datasets, cleaning and split decisions, controlled baseline, DistilBERT and BERT-Tiny configurations, product retrieval algorithm, architecture, UML models, frontend, backend, database, routing logic, evaluation measures, and ethical boundaries. Chapter 4 presents the resulting evidence and interprets it against the research questions.

CHAPTER 4

RESULTS AND DISCUSSION

Model performance, recommendation diagnostics, chatbot experiments, and interpretation

4.1 Dataset and Training Results

The primary cleaning procedure retained 79.9% of the raw intent corpus. Product recommendation and out-of-scope were the largest classes, whereas product_search and product_details were the smallest. Stratification preserved class representation across the training, validation, and test partitions. Figure (2) shows the cleaned class distribution. The imbalance was moderate rather than extreme, but the support difference is sufficient to justify reporting both weighted and macro F1. Figure (3) shows the learning behavior across four epochs. DistilBERT's training loss decreased throughout optimization, while validation loss declined from 0.515 after the first epoch to 0.183 after the fourth. Validation accuracy increased from 0.908 to 0.958 and macro F1 from 0.912 to 0.960. The simultaneous improvement of validation metrics and reduction in validation loss did not indicate late-epoch divergence during the four-epoch run.

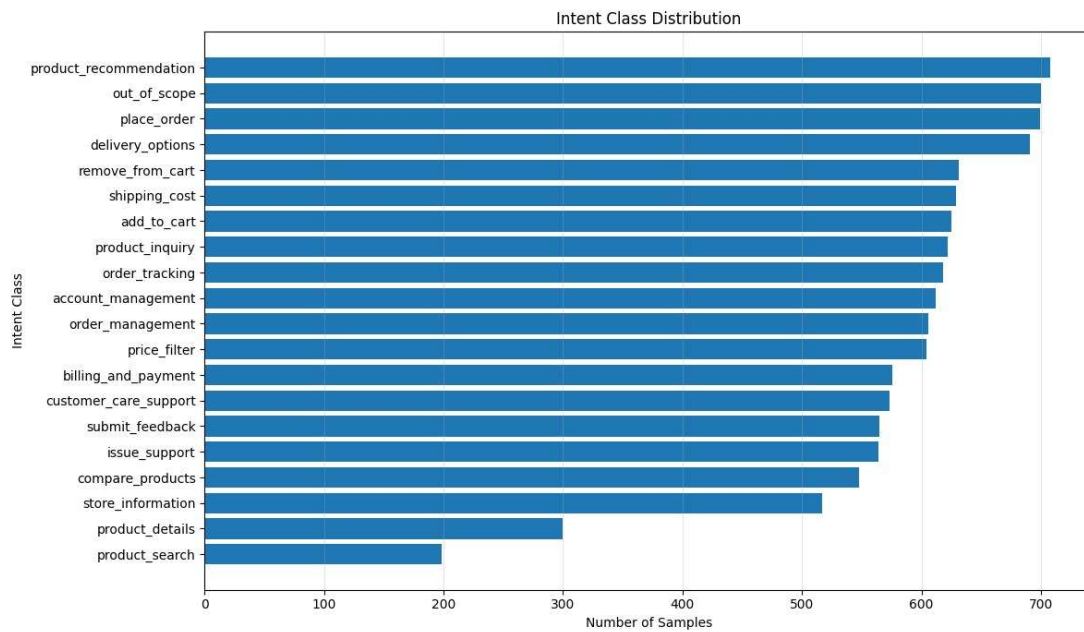


Figure (5): Distribution of the 20 intent classes after primary data cleaning.

The class distribution is moderately imbalanced. Product recommendation and out-of-scope contain comparatively large numbers of examples, while product_search and product_details are smaller. Stratified splitting preserves these proportions across training, validation, and test sets. Weighted and macro metrics are therefore reported together so that overall performance is not interpreted without considering smaller classes.

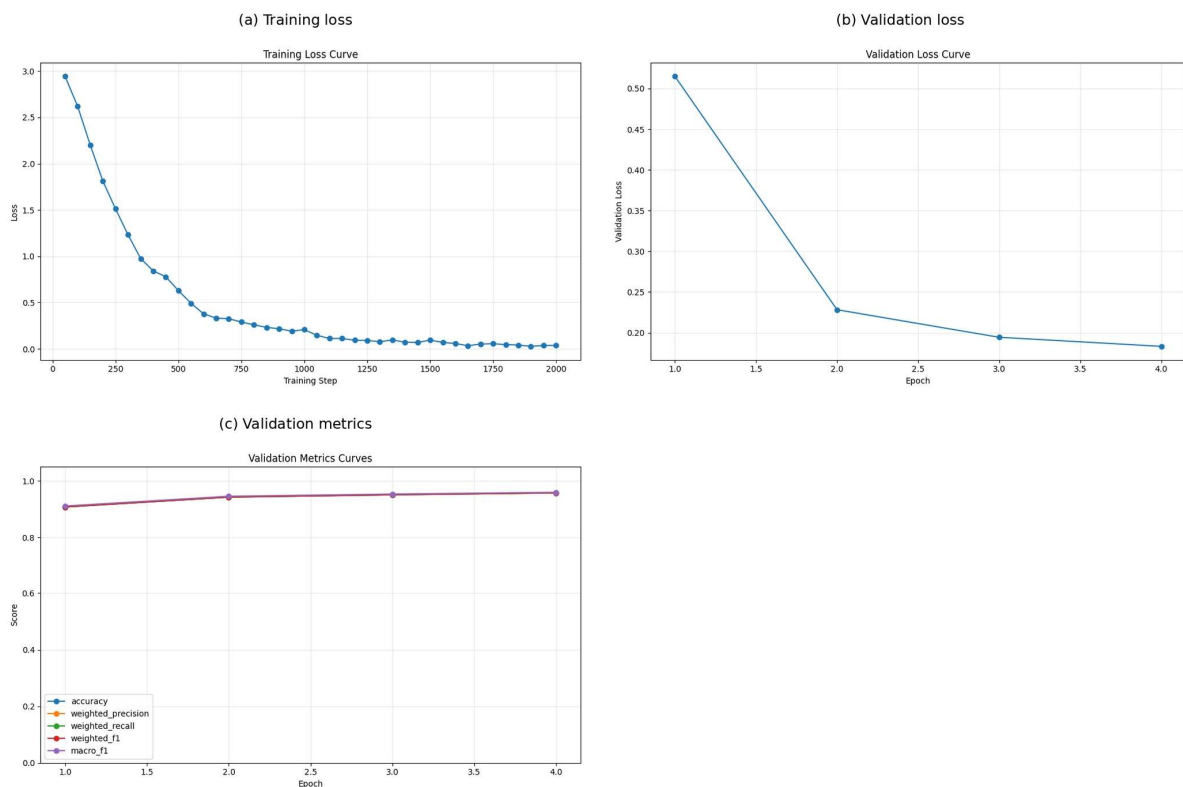


Figure (6): DistilBERT training and validation behavior across four epochs.

Training loss decreased throughout optimization, while validation loss fell from approximately 0.515 after the first epoch to 0.183 after the fourth. Validation accuracy increased from approximately 0.908 to 0.958, and macro F1 increased from approximately 0.912 to 0.960. The simultaneous improvement in validation metrics and reduction in validation loss does not indicate late-epoch divergence during the four-epoch run. However, one fixed seed and one split do not establish stability across repeated trials.

4.2 Controlled Classification Results

On the 1,738-sample primary test set, the TF-IDF logistic-regression baseline achieved 0.8993 accuracy, 0.8978 weighted F1, and 0.9021 macro F1. DistilBERT achieved 0.9563 accuracy, 0.9564 weighted F1, and 0.9595 macro F1. The absolute weighted-F1 improvement was 0.0585. Because both models used the same held-out examples, the comparison supports a controlled performance difference for this corpus. However, the result is based on one fixed random seed and is not accompanied by confidence intervals, repeated-run significance testing, or external validation; it should therefore be interpreted as descriptive evidence for the supplied dataset.

Table (11): Controlled classification performance on the same 1,738-item test set

Model	Accuracy	Weighted precision	Weighted recall	Weighted F1	Macro F1
TF-IDF + logistic regression	0.8993	0.8992	0.8993	0.8978	0.9021
DistilBERT	0.9563	0.9572	0.9563	0.9564	0.9595
Absolute difference	+0.0570	+0.0580	+0.0570	+0.0585	+0.0573

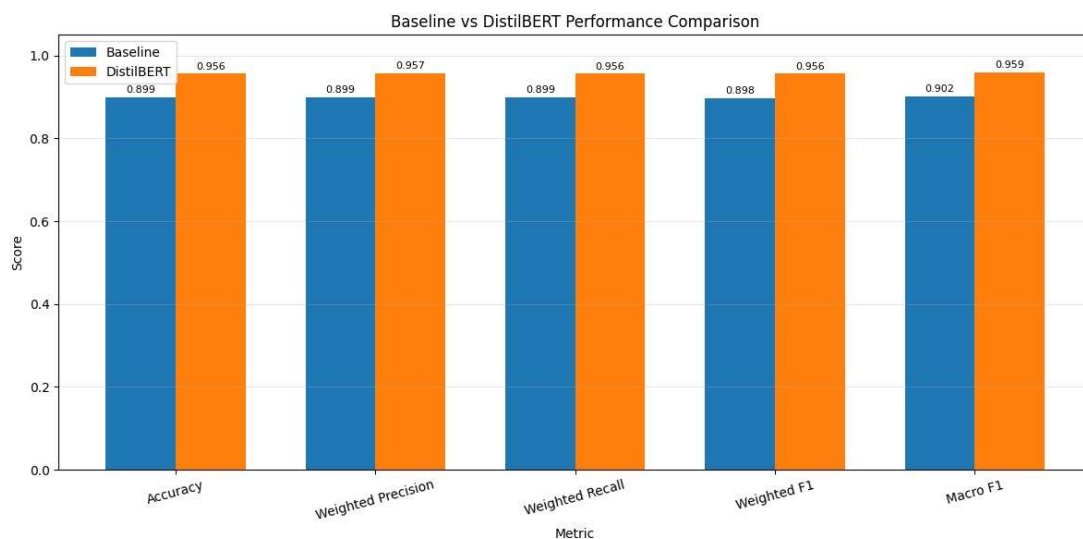


Figure (7): Controlled comparison between TF-IDF logistic regression and DistilBERT.

DistilBERT improves weighted F1 by 0.0585 and accuracy by 0.0570 relative to the controlled baseline. Because both models use the same held-out examples, the result supports the conclusion that contextual representations add value for this corpus. The conclusion remains limited to one fixed-seed experiment and is not accompanied by confidence intervals or repeated-run significance testing.

4.3 Per-Class Analysis

DistilBERT obtained an F1-score of 1.00 for `compare_products`, `price_filter`, `product_details`, `product_recommendation`, and `product_search`. The lowest F1-scores were `out_of_scope` (0.892), `place_order` (0.912), `issue_support` (0.913), `order_tracking` (0.918), and `account_management` (0.928). These lower values occur in categories whose vocabulary can overlap with broader support, payment, order-management, or product-inquiry language. The normalized confusion matrix shows that most observations lie on the diagonal, but the remaining errors are concentrated among semantically related actions rather than being uniformly distributed.

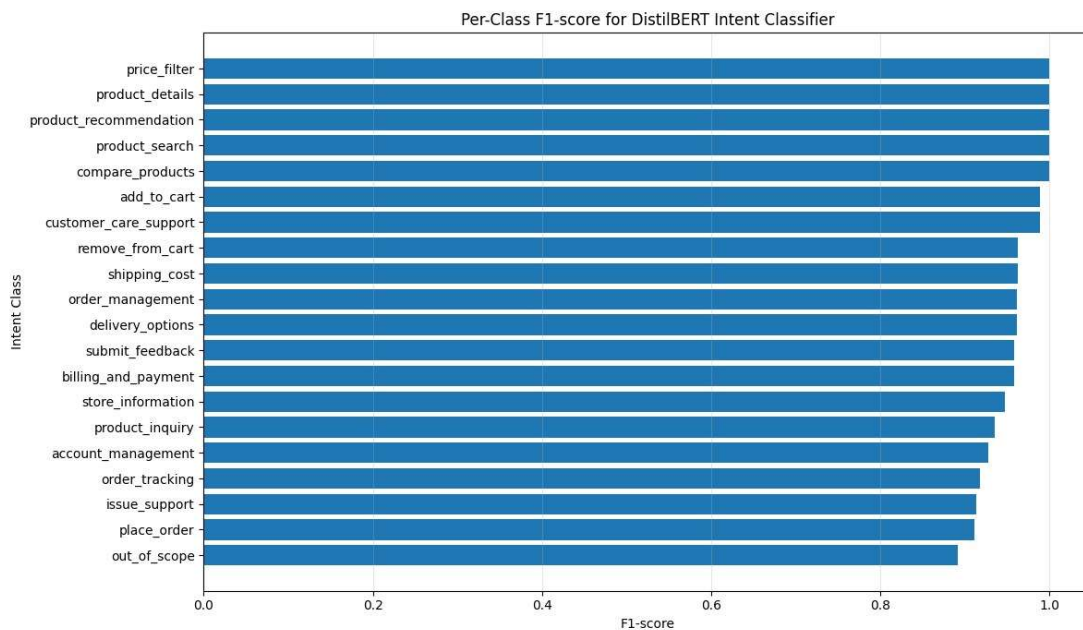


Figure (8): DistilBERT per-class F1-scores across the twenty intents.

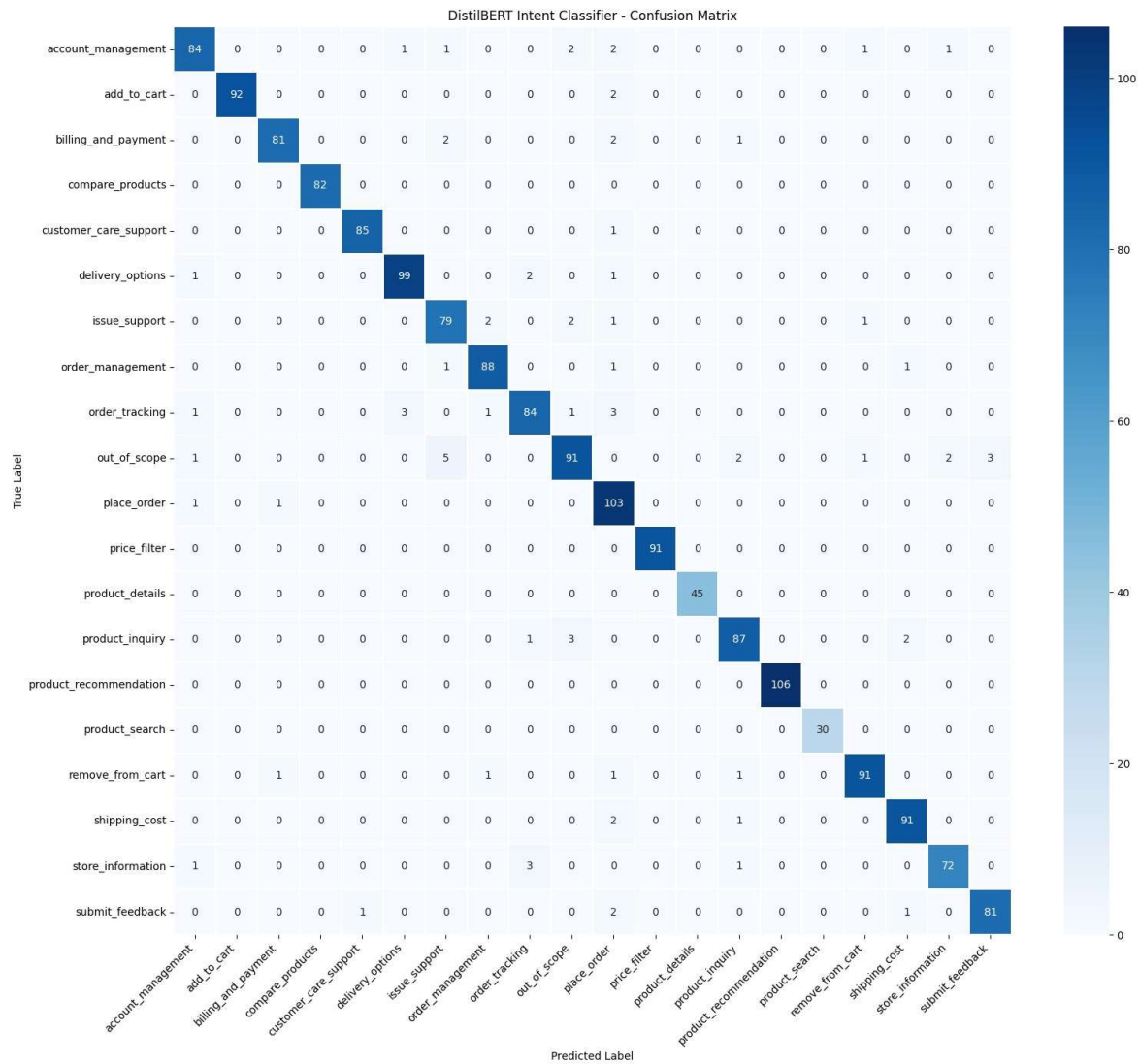


Figure (9): Normalized DistilBERT confusion matrix on the primary test set.

Table (12): Selected class-level DistilBERT results

Intent	Precision	Recall	F1	Interpretation
out_of_scope	0.919	0.867	0.892	Lowest F1; unsupported requests remain difficult
place_order	0.851	0.981	0.912	High recall with lower precision; overlaps service/payment language
issue_support	0.898	0.929	0.913	Confusion with general support-related intents
order_tracking	0.933	0.903	0.918	Overlap with order management and service requests
account_management	0.944	0.913	0.928	Some ambiguity among account and support requests
price_filter	1.000	1.000	1.000	Highly distinctive price expressions
product_recommendation	1.000	1.000	1.000	Distinctive recommendation phrasing
compare_products	1.000	1.000	1.000	Explicit comparison vocabulary

The strongest categories use distinctive lexical patterns, including explicit comparison phrases, recommendation language, product-search wording, and numerical price constraints. Lower-performing categories overlap with broader support, account, payment, and order language. The out_of_scope category remains difficult because unsupported requests are open-ended and can resemble valid store messages. These observations suggest that future work should refine the intent taxonomy, collect hard negative examples, and add clarification behavior for ambiguous predictions.

4.4 Auxiliary BERT-Tiny Results

The auxiliary BERT-Tiny model achieved 0.8548 accuracy, 0.8578 weighted precision, 0.8548 weighted recall, 0.8466 weighted F1, and 0.8563 macro F1 on its separate 1,928-sample test split. Training accuracy improved early, while validation performance peaked before the final epochs and validation loss later increased, indicating that the selected ten-epoch configuration did not provide the same generalization as the primary DistilBERT run. The result demonstrates that the model is deployable and fast, not that its architecture is inherently inferior under controlled conditions. A fair architecture comparison would require identical cleaning, splits, sampling, optimization search, and repeated runs.

Table (13): Auxiliary BERT-Tiny results

Metric	Value	Qualification
Test loss	0.7399	Separate 1,928-item test split
Accuracy	0.8548	Not directly comparable with the primary experiment
Weighted precision	0.8578	Training was capped at 200 cases per intent
Weighted recall	0.8548	Different cleaning and split
Weighted F1	0.8466	Reported as auxiliary deployment evidence
Macro F1	0.8563	Twenty intent labels

BERT-Tiny achieved 0.8548 accuracy and 0.8466 weighted F1 on its separate 1,928-item test set. The model is small and deployable, but its result cannot be interpreted as evidence that the architecture is inherently weaker than DistilBERT. A fair comparison would require identical cleaning, sampling, data splits, optimization budgets, and repeated runs.

4.5 Recommendation Diagnostics

The recommendation engine was evaluated through selected retrieval queries.

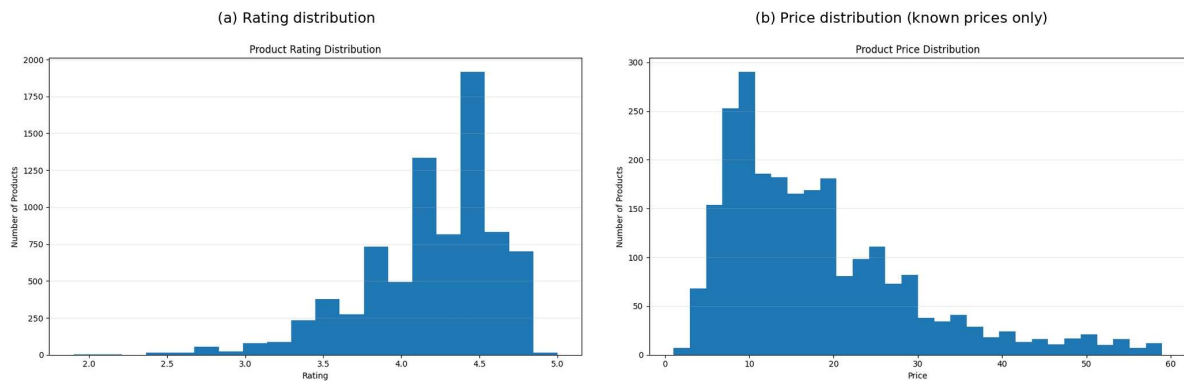


Figure (10): Product catalog price and rating distributions.

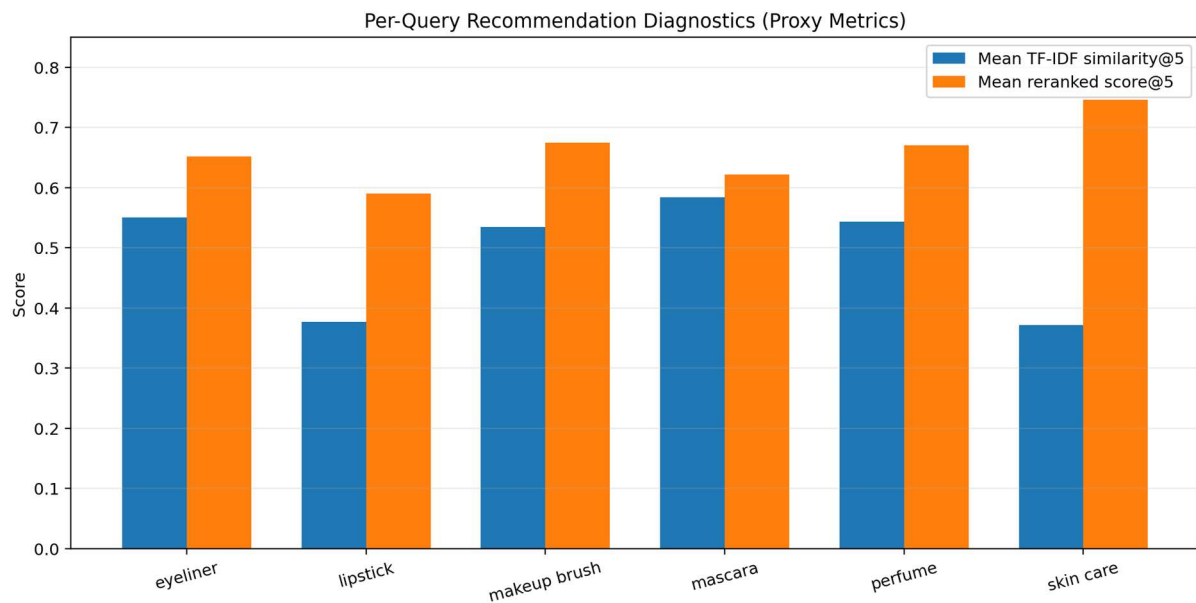


Figure (11): Recommendation retrieval diagnostics for six representative queries.

Table (14): Recommendation diagnostics at $k = 5$

Query	Items returned	Mean similarity@5	Mean reranked score@5
eyeliner	5	0.5506	0.6516
lipstick	5	0.3768	0.5900
makeup brush	5	0.5343	0.6748
mascara	5	0.5840	0.6217
perfume	5	0.5432	0.6704
skin care	5	0.3713	0.7461
Overall	30	0.4934	0.6591

The engine returned five products for all six selected queries, producing coverage@5 of 1.0 under the project's operational definition. Mean similarity@5 was 0.4934 and mean

reranked score@5 was 0.6591. Mascara produced the highest average lexical similarity, while lipstick and skin care produced lower values. These results verify that the service returns and ranks candidates; they do not measure user satisfaction or precision against human relevance labels.

Qualitative inspection exposed several limitations. Lexical similarity may return accessories or brushes that share vocabulary with the requested product, duplicate titles may appear, and semantically related products may be missed when the query and product description use different terms. Missing price data further reduces budget-request coverage. Semantic embeddings, product-type classifiers, duplicate removal, and human relevance assessment are required for stronger recommendation evaluation.

4.6 Chatbot and API Experiments

4.6.1 Executed Chatbot Scenarios

DistilBERT's mean confidence across the selected scenarios was 92.0%, compared with 82.0% for BERT-Tiny. Median recorded latency was 0.066 seconds for DistilBERT and 0.011 seconds for BERT-Tiny in the local test sequence. The latency figures include the project's runtime environment and model cache state; they are not hardware-independent benchmarks. Product-returning scenarios generated 26 product cards per model across the set. Policy and support scenarios returned explanatory text without product cards, which is consistent with the intended route design.

Table (15): Executed DistilBERT chatbot scenarios, part A

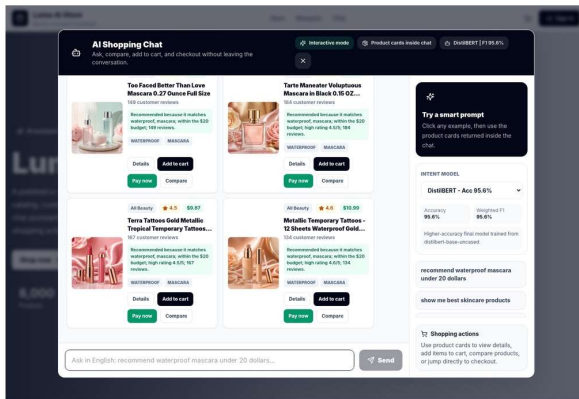
ID	User request	Raw intent → final route	Confidence	Observed output
CT-01	Waterproof mascara under \$20	price_filter	98.6%	6 product cards
CT-02	Niacinamide serum for dry skin	product_search	47.2%	6 product cards
CT-03	Compare two eyeliners	compare_products	98.8%	2 product cards
CT-04	Shipping cost	shipping_cost	99.2%	Policy/support response
CT-05	Track ORD-123456	order_tracking	99.1%	Policy/support response
CT-06	Damaged product support	issue_support	99.1%	Policy/support response

Table (16): Executed DistilBERT chatbot scenarios, part B

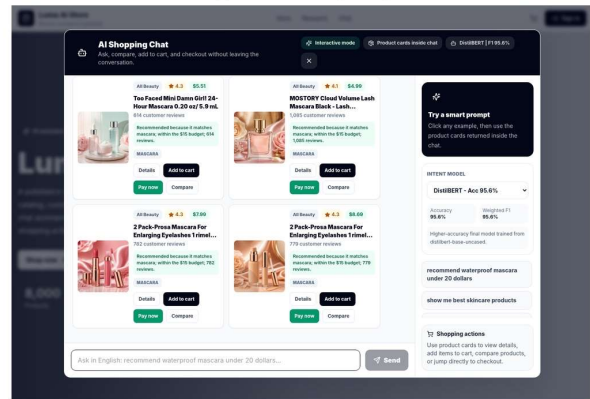
ID	User request	Raw intent → final route	Confidence	Observed output
CT-07	Visa payment	billing_and_payment	94.4%	Policy/support response
CT-08	Football joke	out_of_scope	99.5%	Policy/support response

ID	User request	Raw intent → final route	Confidence	Observed output
CT-09	Misspelled mascara under \$15	remove_from_cart → price_filter	80.5%	6 product cards
CT-10	Oral-B toothbrushes	product inquiry	90.7%	6 product cards
CT-11	Checkout request	billing_and_payment → place_order	97.7%	Policy/support response
CT-12	Password reset	account management	99.2%	Policy/support response

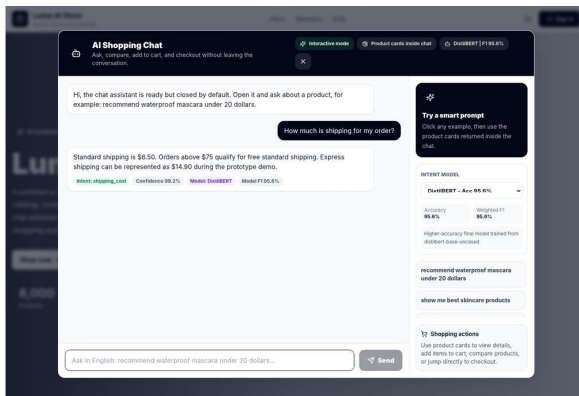
(a) Price-filtered recommendation



(b) Typo-tolerant budget request



(c) Shipping policy request



(d) Out-of-scope request

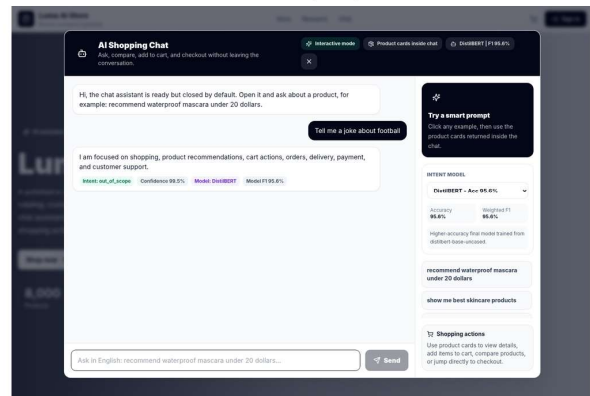


Figure (12): Executed chatbot tests for product, typo, policy, and out-of-scope requests.

For each deployed model, the raw classifier intent matched the final application route in ten of twelve cases (83.3%). The two changed cases were the typo-heavy budget request and the checkout request. DistilBERT predicted `remove_from_cart` for the misspelled mascara message before the explicit budget rule selected `price_filter`. It predicted `billing_and_payment` for “I want to checkout” before the checkout rule selected `place_order`. These cases demonstrate the operational value of deterministic refinement and the importance of reporting the raw prediction rather than attributing complete scenario success to the classifier alone.

The selected DistilBERT scenarios recorded a mean confidence of approximately 92.0%, compared with approximately 82.0% for BERT-Tiny. Median local latency was approximately 0.066 seconds for DistilBERT and 0.011 seconds for BERT-Tiny. These

values were collected sequentially in one runtime with different response paths and cache states; they are not hardware-independent benchmarks.

4.6.2 Automated API Verification

Table (17): Automated application verification

Test group	Representative checks	Result	Interpretation
Availability and data	Health, model list, catalog and category retrieval	Passed	Core services and local data were accessible
Intelligent functions	Search, recommendation, price filtering, comparison	Passed	Expected response structures and product cards were returned
Business scenarios	Shipping, payment, tracking, support, out-of-scope	Passed	Expected policy or support routes were returned
Total	15 automated assertions	15/15 passed	Functional verification only; not production certification

The final verification evidence consists of two complementary layers. The documented regression script executed 52 chatbot cases covering direct requests, typographical errors, context-dependent actions, result limits, product details, comparison, cart behavior, shipping, payment, tracking, support, and out-of-scope inputs; all 52 cases passed. A second focused validation suite executed 15 checks covering service health, the 8,000-record catalog, product categories, local loading of DistilBERT and BERT-Tiny, budget-constrained recommendation, product-type relevance, comparison uniqueness, typographical-error recovery, shipping responses, negative budgets, ambiguous requests, out-of-catalog requests, blank messages, unsupported model keys, and overlong messages; all 15 checks passed. These results establish reproducible functional behavior in the evaluated local environment, but they do not constitute production certification.

4.6.3 Interface Verification

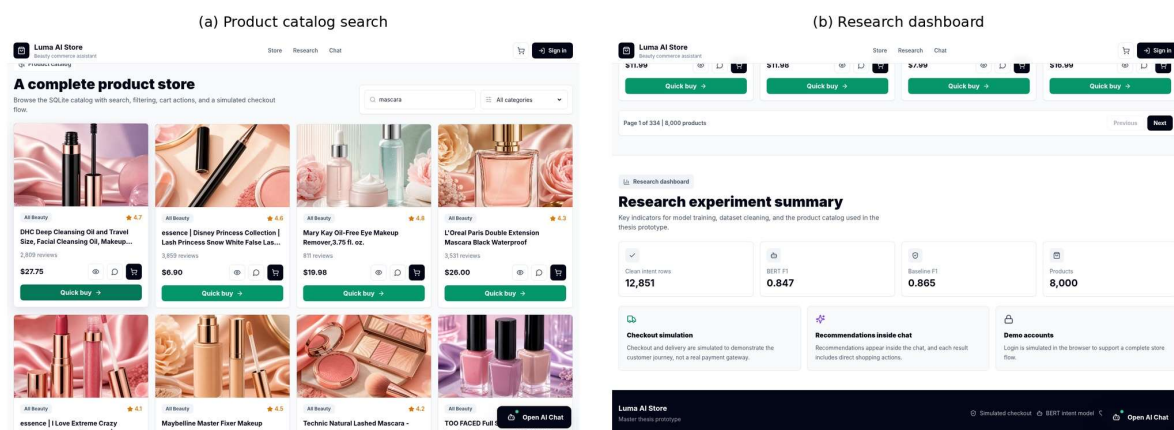


Figure (13): Running home, catalog-search, and product interfaces.

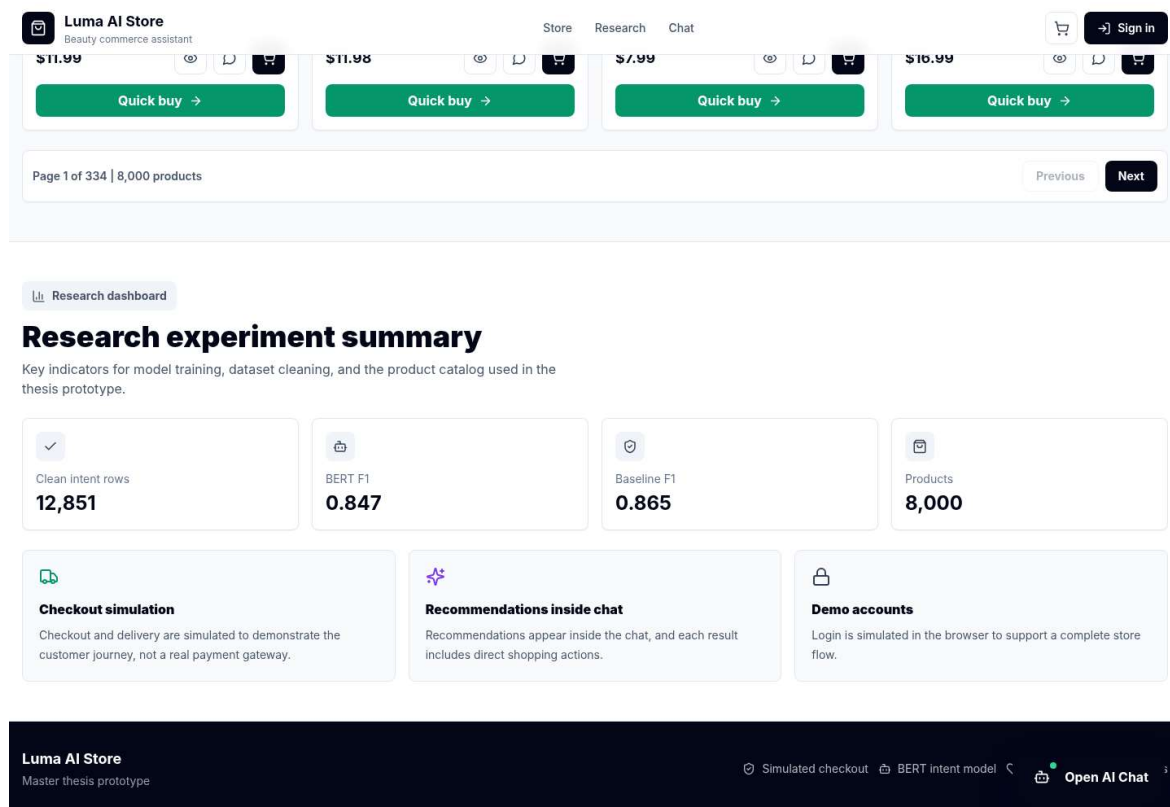


Figure (14): Research dashboard and evaluation interface.

The interface screenshots confirm that users can access the landing page, search the catalog, open the chat interface, view returned product cards, and inspect research metrics. The dashboard exposes model information and evaluation artifacts, making the project suitable for demonstration and examination. Visual evidence complements but does not replace functional assertions or model metrics.

4.6.4 Final Runtime Verification and Hidden-Issue Corrections

The final project archive was executed again from its packaged backend, trained DistilBERT and BERT-Tiny models, SQLite catalog, local product assets, and frontend bundle. The final visual revision restored the approved identity observed in the original project: a dark navy navigation system combined with cream, peach, soft rose, and warm gold product accents. The hero no longer depends on an internet URL and uses a packaged beauty-product asset. The review focused on visible failures and hidden logical defects that could appear during a fresh start, invalid input, recommendation ranking, product comparison, typographical errors, unavailable prices, or requests outside the beauty catalog. A correction was accepted only after the affected behavior was re-executed and included in the 52-case regression suite or the focused 15-check runtime audit.

Table (18): Final implementation defects, corrections, and verification evidence

Detected issue	Correction implemented	Verification evidence
Hero depended on an external image URL and did not reliably display offline.	Replaced the external dependency with a packaged beauty-product hero asset and referenced it from the source and built frontend.	The approved-identity homepage loaded with the local hero while the environment was offline.

Detected issue	Correction implemented	Verification evidence
The runtime failed when Transformers was unavailable.	Added a project-local PyTorch and safetensors inference runtime for DistilBERT and BERT-Tiny.	Both models returned valid predictions; runtime backend reported project-local-pytorch.
Database-reading services could be imported before schema initialization.	Initialized the SQLite schema before router/service imports and added an empty-catalog guard.	Fresh startup and health/catalog checks passed.
Chat and recommendation payloads accepted invalid or unbounded values.	Added whitespace, length, model-key, and recommendation-limit validation.	Invalid requests returned HTTP 422.
Catalog LIKE queries treated % and _ as wildcards.	Escaped user search patterns and added query-length limits.	Literal wildcard search no longer returned the complete catalog.
A declared evaluation route referred to a missing artifact.	Returned the packaged evaluation summary with an explicit CPU-runtime note when no separate file exists.	All evaluation endpoints returned HTTP 200.
Raw chat logs could retain e-mail addresses and order identifiers.	Added targeted redaction before writing research logs.	Stored log replaced identifiers with [email] and [order-id].
Price-filter title preference treated an attribute as a product type.	Removed waterproof from product-type matching while retaining it as a query attribute.	Returned price-filter titles were mascara/lash products within the requested budget.
The database utility imported an excluded data-engineering module.	Replaced it with a packaged-database validation utility.	Utility reported 8,000 available products without import errors.
Development API configuration and preview CORS were too narrow.	Added an environment-based API base and allowed Vite development and preview origins.	Source inspection and frontend execution confirmed local API access.
Ranked recommendations could display duplicate normalized titles.	Added post-ranking visible-title deduplication while preserving the strongest-ranked record.	The eyeliner validation returned six results with six unique visible titles.
Negative and zero budgets were accepted or interpreted inconsistently.	Added explicit positive-price validation before retrieval.	Requests below -5 and 0 returned controlled validation messages and no products.
Textual, malformed, and unrealistic budgets were not handled consistently.	Added parsing for common written numbers and rejection for malformed or values above \$10,000.	“twenty” was parsed as 20; \$abc and 999999 were rejected with specific guidance.
The frontend generated deterministic estimated prices when catalog prices were missing.	Removed the estimated-price fallback, displayed “Price unavailable,” and blocked cart/checkout actions without a verified price.	Source and built bundle contain the corrected behavior and no “Est.” price fallback.
Generic API failures exposed raw response text rather than a useful validation message.	Parsed FastAPI detail fields and preserved a separate connection-unavailable message.	Error-state screenshots show understandable validation and recovery messages.
A descriptive attribute could dominate lexical retrieval even when it did not identify the requested product type.	Separated product-type anchors from descriptive attributes and retained rating/popularity reranking.	The waterproof-mascara test returned mascara products only and excluded unrelated waterproof items.
The final built frontend did not initially include the last source-level price-integrity correction because package download failed.	Patched the supplied prebuilt bundle to match the verified source behavior and retained the source as the rebuild reference.	The final bundle contains “Price unavailable,” blocks unpriced cart additions, and passes the asset/bundle validation checks.

Detected issue	Correction implemented	Verification evidence
The project did not preserve a single human-readable map of success and error screenshots.	Added qa_screenshots/SCREENSHOT_MANIFEST.md and grouped evidence by successful execution and validation/recovery states.	The final archive contains eight success images and eight error/recovery images.
The redesigned interface did not match the visual identity shown in the original product screenshots.	Restored the approved navy, cream, peach, soft-rose, and warm-gold identity and aligned buttons, cards, navigation, and chat surfaces.	The final homepage screenshot documents the approved identity in the running application.
Product-card images could be unrelated to the visible product title when chosen by card position.	Mapped local images according to normalized product type and title rather than display order.	Executed recommendation and comparison screenshots show category-consistent beauty-product imagery.

The final audit confirmed 8,000 product records, two locally accessible trained model packages, a packaged hero asset, correct product-image mapping for the tested categories, budget-compliant price-filter results, unique visible titles in recommendation and comparison outputs, and no failed cases in either final suite. Missing prices remain a declared data limitation. The corrected interface displays “Price unavailable” and blocks cart or checkout actions for records without a verified price instead of fabricating an estimated amount. The final recommendation logic also preserves a product-type anchor so that a descriptive word such as “waterproof” cannot cause unrelated product types to outrank the requested mascara category.

The following screenshots were captured from the final running application after the approved visual identity, local hero asset, recommendation logic, duplicate-title removal, product-type anchoring, and validation messages had been implemented. The images are presented at readable width so that interface labels, returned product cards, and error messages can be inspected directly.

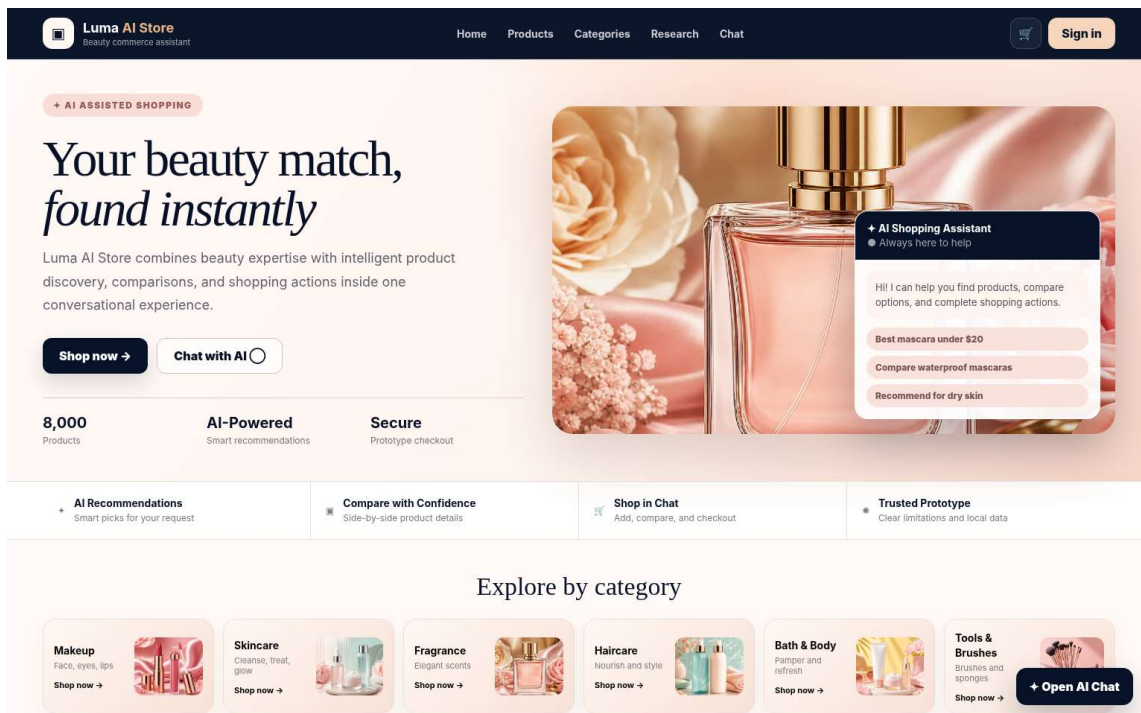


Figure (15): Final homepage using the approved beauty-store identity and packaged local hero asset.

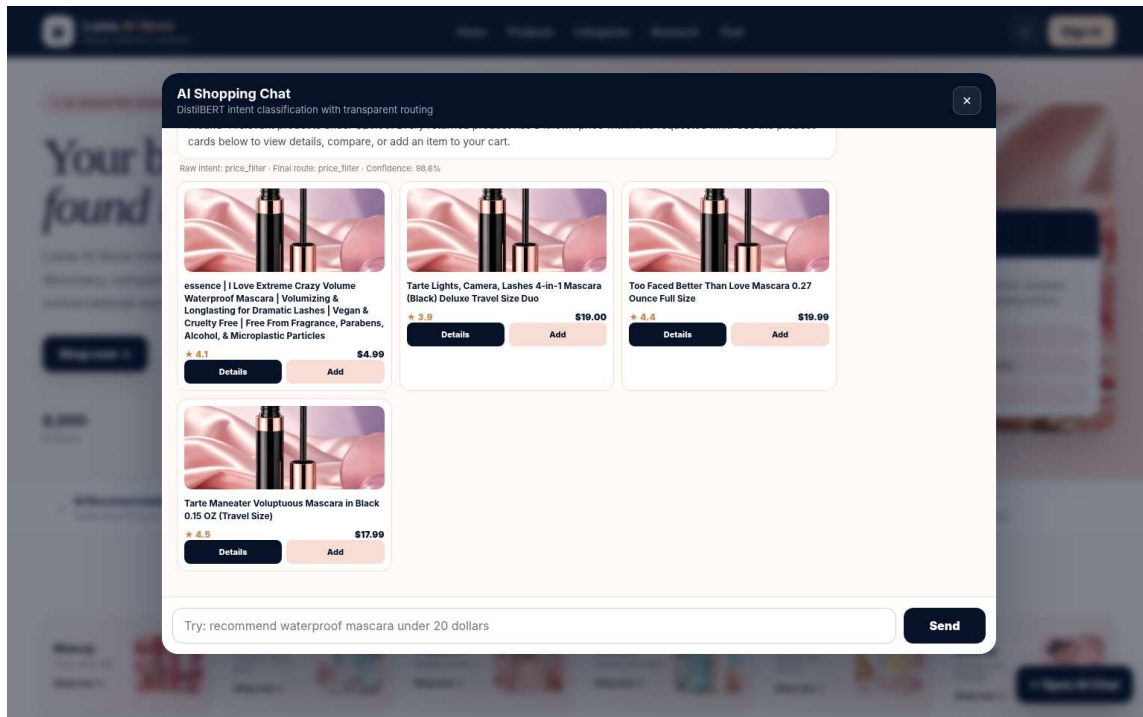


Figure (16): Executed DistilBERT price-constrained recommendation with budget-compliant, product-type-relevant results.

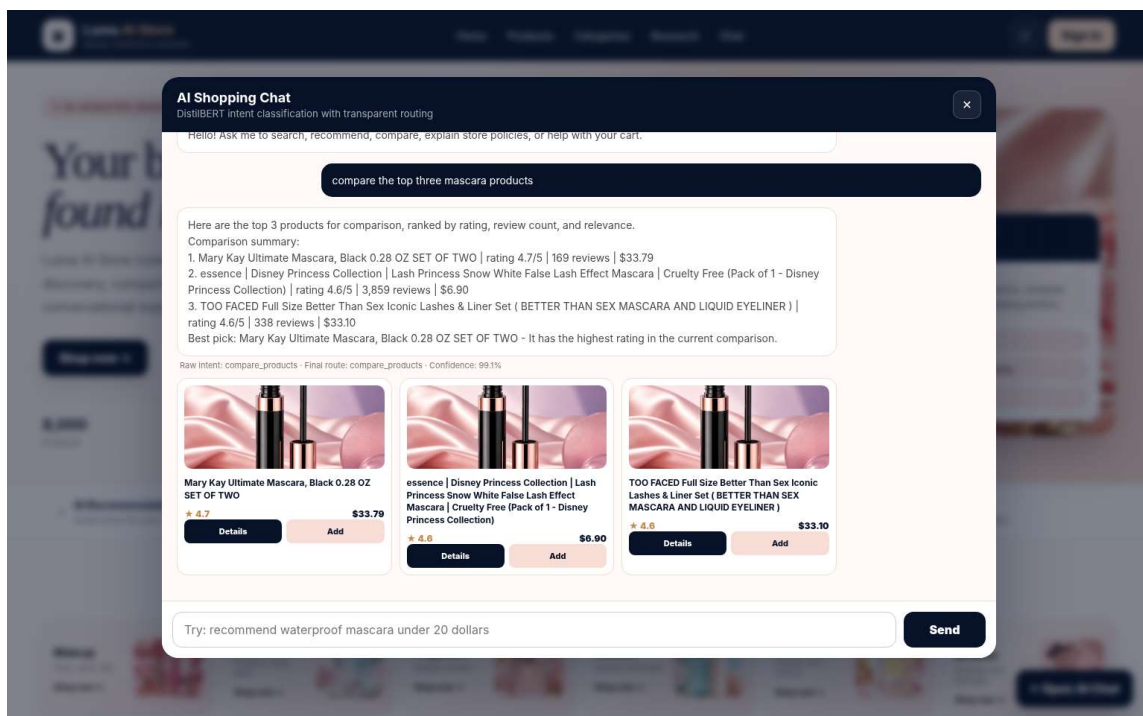


Figure (17): Executed product-comparison scenario with distinct visible titles and category-consistent images.

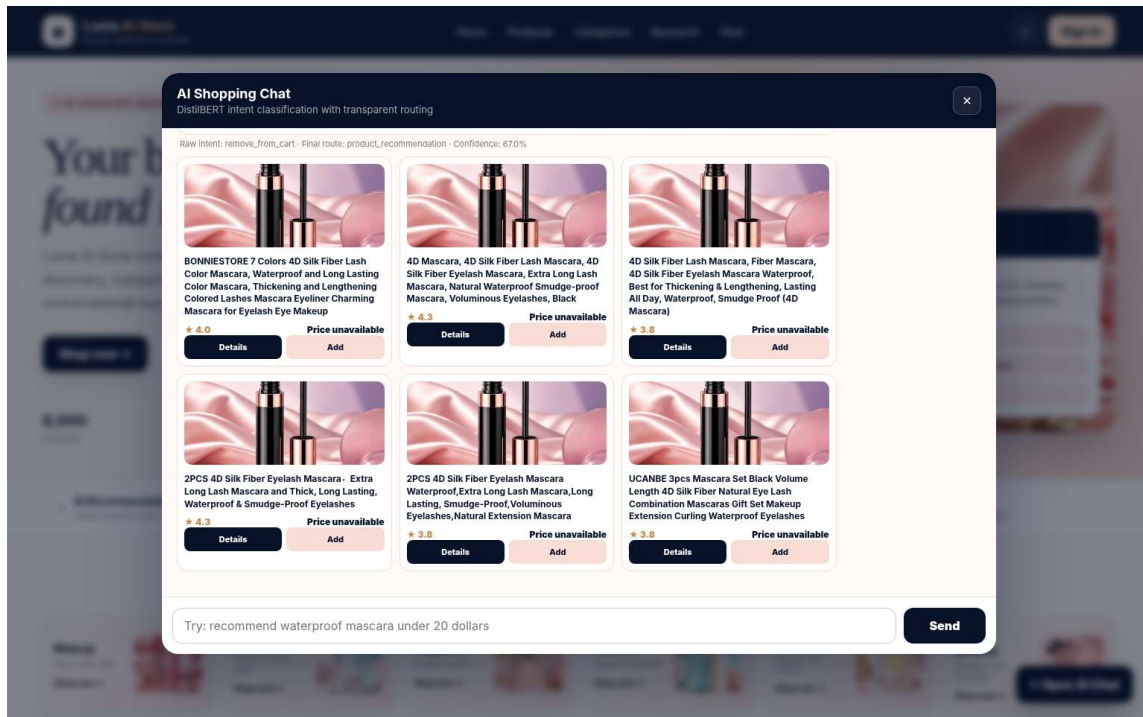


Figure (18): Executed typographical-error recovery showing hybrid normalization and final routing behavior.

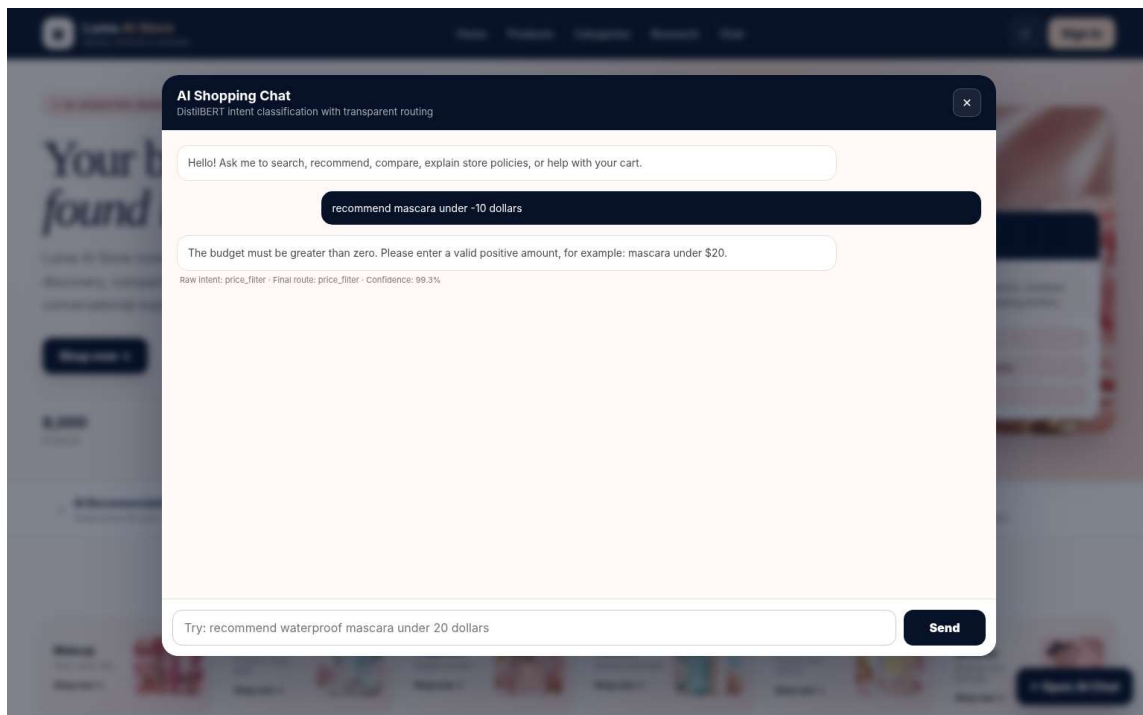


Figure (19): Controlled negative-budget validation message with no product retrieval.

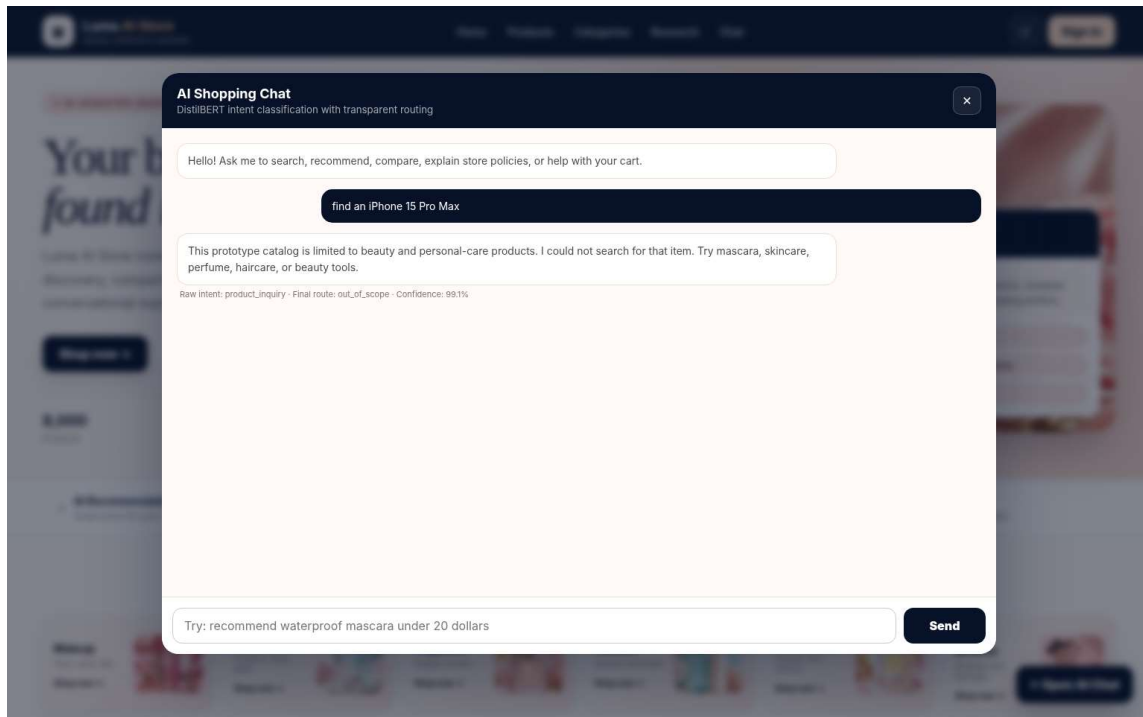


Figure (20): Controlled out-of-catalog response that avoids returning unrelated beauty products.

Figures (15)–(20) distinguish successful operations from controlled validation and recovery behavior. Figures (15)–(18) document the approved interface identity, price-constrained recommendation, comparison, and typographical-error recovery. Figures (19)–(20) document expected responses to a negative budget and a product request outside the catalog. These error states are not unresolved crashes; they show that invalid or unsupported inputs are stopped before unrelated products or unsafe shopping actions are produced.

4.7 Discussion of Research Questions

Table (19): Research-question findings

Research Question	Answer Supported by the Results
RQ1	DistilBERT achieved 0.9563 accuracy and 0.9564 weighted F1, exceeding the controlled TF-IDF baseline by 0.0570 accuracy and 0.0585 weighted F1.
RQ2	The most difficult classes included out_of_scope, place_order, issue_support, order_tracking, and account_management; errors concentrate among semantically related service and transactional intents.
RQ3	The deployed system uses a hybrid path in which the model supplies a raw intent and confidence, while deterministic safeguards, extracted constraints, and business policies select the final route.
RQ4	The twelve scenarios and fifteen automated assertions demonstrate selected functional coverage. Two scenarios required routing overrides, confirming that functional success cannot be counted as additional classifier accuracy.
RQ5	The recommendation engine returns ranked candidates, but the evidence is limited to proxy diagnostics. Missing prices, absence of user histories, and lack of human relevance judgments prevent claims of personalization or user satisfaction.

The controlled results answer RQ1. As shown in Table (4) and Figure (4), DistilBERT substantially improved classification over the TF-IDF logistic-regression baseline on the same held-out test set. The weighted-F1 gain of 0.0585 indicates that contextual representations added value beyond unigram and bigram statistics for this corpus. The conclusion is limited to the supplied data and a single fixed-seed experiment; external validity requires new datasets and repeated trials.

RQ2 is answered by the class-level analysis in Figure (5) and Table (5). Perfect or near-perfect scores occurred for intents with distinctive expressions, including price filtering and explicit product actions. Lower scores occurred for out_of_scope, place_order, issue_support, tracking, and account-management utterances, where vocabulary overlaps with broader service requests. This pattern indicates that future improvement should prioritize more diverse negative examples, ambiguous utterances, clarification strategies, and calibrated confidence thresholds.

RQ3 concerns the transition from prediction to behavior. Figure (1) shows that the implemented architecture uses a hybrid pipeline: the model supplies a probability-based intent, while explicit normalization and routing rules protect high-value business actions and common noisy expressions. Tables (8) and (9) record both raw and final intents. The approach is auditable because the API records the raw prediction, final route, and reason for any override; however, the existence of overrides means that final application behavior cannot be used as an additional measure of classifier accuracy.

RQ4 is answered by the executed scenarios and API report. Tables (8) and (9), Figure (7), Table (10), and Figure (8) show that the chatbot returned the expected response type for product discovery, budget filtering, comparison, policy questions, order tracking, damaged-item support, payment, account management, and out-of-scope messages. These results demonstrate functional coverage under the selected test conditions, but they are not a substitute for randomly sampled real conversations, human judgments, error-severity analysis, stress testing, or long-term reliability measurement.

RQ5 concerns limits of recommendation evidence. Table (7) and Figure (6) show that the content-based engine consistently returned candidates and produced measurable similarity and reranking scores. These diagnostics verify retrieval behavior only. Without user histories, human relevance judgments, or interaction outcomes, they cannot establish personalization, recommendation usefulness, diversity, or satisfaction. Missing price values also constrain price-based selection to the subset with documented prices.

The selected chatbot cases also reveal a practical design trade-off. Deterministic overrides improved task completion for misspelled price requests and checkout language, but they can mask classifier errors if only the final route is reported. Both values must therefore be retained for auditability. During deployment, the override rate should be monitored as a separate operational indicator: an increasing rate may signal model drift, a change in user language, inadequate training coverage, or routing rules that have become too broad.

The observed latency values should be interpreted cautiously. They were collected sequentially in one local runtime and include different response paths; they are not controlled throughput benchmarks. DistilBERT's higher mean confidence and slower median latency are consistent with its larger capacity relative to BERT-Tiny, but hardware-independent conclusions would require warm-up control, repeated measurements, identical batches, concurrency tests, and confidence calibration.

4.8 Limitations and Threats to Validity

Internal validity is limited by the use of one fixed seed and one split in the controlled experiment. The observed difference between the baseline and DistilBERT is descriptive and requires repeated runs for confidence intervals and significance analysis. BERT-Tiny was trained through a different pipeline, so its result cannot support architectural ranking.

Construct validity is limited by the available evaluation evidence. Functional completion is based on purposefully selected scenarios rather than a random sample of user conversations. Recommendation metrics measure retrieval behavior rather than perceived relevance, diversity, usefulness, or satisfaction. Confidence values are model outputs and have not been calibrated as probabilities of correctness.

External validity is limited by the English-language intent corpus, the beauty-focused product catalog, undocumented intent-data provenance, missing product prices, and the absence of production traffic. The results should not be generalized to Arabic, multilingual commerce, other product domains, real payment systems, or high-load deployment without additional evidence.

Implementation validity was strengthened by adding bounded dependency ranges, an offline PyTorch and safetensors runtime for both packaged Transformer models, early database-schema initialization, message and model-key validation, budget validation, escaped catalog search patterns, privacy-oriented log redaction, duplicate-title suppression, verified-price handling, an environment-configurable API base, and a local hero asset. The final

release includes source code, a tested prebuilt frontend, automated reports, execution logs, and success/error screenshots. Remaining implementation limits include browser-local simulated authentication, simulated payment and shipping, and the absence of production-scale security, accessibility, concurrency, and long-duration testing.

4.9 Chapter Summary

This chapter showed that DistilBERT substantially outperformed the controlled TF-IDF baseline on the supplied intent dataset, while BERT-Tiny provided separate lightweight deployment evidence. It identified class-level difficulties, demonstrated the operation and limitations of the content-based retrieval engine, and verified selected chatbot and API functions. The discussion answered the research questions while preserving the distinction between model accuracy, routing behavior, retrieval diagnostics, and software execution.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Conclusions, contributions, recommendations, and planned extensions

5.1 Conclusion

Overall, the study demonstrates a complete and inspectable research implementation. Its strongest evidence is the controlled DistilBERT comparison and the traceable integration between classification and application actions. Its principal weaknesses are undocumented provenance for the intent corpus, lack of repeated model runs, a small selected functional test set, absence of user-item relevance labels, and no user or production-level evaluation. Acknowledging these limits provides a more credible basis for subsequent development than combining heterogeneous outputs into a single performance claim.

The research therefore answers the central problem with a qualified conclusion: a compact Transformer can provide strong intent classification within a functioning e-commerce application, but dependable application behavior emerges from the combination of classification, transparent routing rules, catalog retrieval, database-backed actions, and interface feedback. The primary evidence supports DistilBERT over the controlled TF-IDF baseline for the supplied dataset. It does not establish superiority over BERT-Tiny because that model was trained through a different pipeline. The recommendation component remains content-based and non-personalized because the project contains no user-item histories or human relevance judgments; consequently, the study does not establish real-user recommendation quality, satisfaction, or production readiness.

The project meets its central implementation objective by connecting a trained intent classifier to a working e-commerce prototype. A user can submit a natural-language request, the backend can produce and record a raw intent, routing logic can refine the decision when explicit operational evidence is present, the appropriate product or business service can execute, and the frontend can render a structured result. The full path is supported by source code, trained models, a local database, metrics, automated tests, and screenshots.

The strongest empirical result is the controlled comparison between DistilBERT and the TF-IDF logistic-regression baseline. DistilBERT achieved 0.9564 weighted F1 compared with 0.8978 for the baseline on the same 1,738 held-out examples. The auxiliary BERT-Tiny result demonstrates that a smaller model can be deployed, but it does not establish an architectural comparison because the data pipeline differs. The recommendation engine remains a content-based retrieval prototype, and its diagnostics do not establish personalized recommendation quality or user satisfaction.

5.2 Summary of Contributions

- A controlled comparison between TF-IDF logistic regression and DistilBERT on the same cleaned dataset and the same held-out test split.
- A transparent distinction between the primary DistilBERT experiment and the separate BERT-Tiny pipeline.
- An implemented full-stack e-commerce chatbot that connects intent prediction with search, recommendation, policy, cart, order, and support functions.
- An auditable distinction between raw model intent and final application route, including deterministic override reasons.
- A three-layer evaluation protocol comprising model metrics, recommendation diagnostics, and executed software scenarios.
- A documented analysis of limitations involving dataset provenance, catalog completeness, lack of human relevance judgments, and absence of production testing.

A further practical contribution is the proposal to monitor override rate during deployment. A rising rate would indicate that user language has shifted, the model no longer covers common requests, or the rule layer is compensating for systematic errors. Monitoring raw intent, final route, confidence, retrieval failure, and user correction events separately would provide a clearer operational picture than a single aggregate success measure.

5.3 Recommendations

5.3.1 Data

Create a formal data card documenting the source of the intent corpus, annotation procedure, label definitions, annotator agreement, licensing, consent, and privacy review. Collect real, consented, and anonymized conversations, with particular attention to spelling errors, paraphrases, follow-up references, and unsupported requests. Improve product normalization, duplicate detection, category labels, and price completeness.

5.3.2 Modeling

Repeat the primary experiment across multiple random seeds and report mean performance, standard deviation, confidence intervals, and paired error analysis. Train DistilBERT and BERT-Tiny on identical preprocessing, splits, sample sizes, and tuning budgets before making an architectural comparison. Add confidence calibration and clarification questions for ambiguous or low-confidence inputs.

5.3.3 Recommendation

Obtain human relevance judgments and evaluate precision, ranking quality, diversity, novelty, usefulness, and failure severity. Investigate semantic sentence and product embeddings or hybrid lexical-semantic retrieval to reduce synonym and lexical-mismatch errors. Multi-turn preference elicitation should be evaluated only after representative interaction data are available.

5.3.4 Deployment

Conduct structured usability, accessibility, security, privacy, load, concurrency, recovery, and long-duration reliability testing before public deployment. Implement authentication, authorization, secure secret management, audit logging, rate limiting, monitoring, backup, and incident procedures. Simulated payment and shipping functions must be replaced with verified integrations before any commercial use.

5.3.5 Multilingual Extension

Extend the system to Arabic and multilingual commerce only after collecting representative labeled data and defining language-specific intent taxonomies, tokenization procedures, dialect coverage, code-mixed test sets, and human evaluation protocols. Arabic

support should not be inferred from the presence of an Arabic abstract because the current classifier and tests are English-only.

5.4 Future Work Roadmap

Future-work roadmap

Horizon	Area	Planned Work
Short term	Reproducibility	Publish a locked environment, data card, model card, and versioned test suite.
Short term	Data quality	Complete product prices, normalize titles, remove duplicates, and validate categories.
Medium term	Model robustness	Run repeated experiments, calibrate confidence, and add clarification for ambiguous requests.
Medium term	Recommendation quality	Collect relevance judgments and compare lexical, semantic, and hybrid retrieval.
Medium term	User evaluation	Conduct task-based usability and satisfaction studies with consented participants.
Long term	Production engineering	Implement secure identity, real integrations, monitoring, load testing, recovery, and governance.
Long term	Multilingual capability	Collect Arabic and code-mixed data and evaluate language-specific models and taxonomies.

5.5 Final Statement

The study demonstrates that a compact Transformer can provide strong intent classification inside an operational e-commerce prototype, but dependable application behavior is produced by the combined effect of classification, explicit routing, product retrieval, database services, and interface feedback. The value of the work lies in both the implemented artifact and the disciplined separation of evidence. This separation provides a credible foundation for future research and development without overstating personalization, production readiness, or generalization beyond the evaluated conditions.

References

- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749. <https://doi.org/10.1109/TKDE.2005.99>
- Aly, W. M., Eskaf, K. A., & Selim, A. S. (2017). Fuzzy mobile expert system for academic advising. In 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1–5). IEEE. <https://doi.org/10.1109/CCECE.2017.7946846>
- Deriu, J., Rodrigo, A., Otegi, A., Echegoyen, G., Rosset, S., Agirre, E., & Cieliebak, M. (2021). Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54, 755–810. <https://doi.org/10.1007/s10462-020-09866-x>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 4171–4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- Gao, C., Lei, W., He, X., de Rijke, M., & Chua, T.-S. (2021). Advances and challenges in conversational recommender systems: A survey. *AI Open*, 2, 100–126. <https://doi.org/10.1016/j.aiopen.2021.06.002>
- Hou, Y., Li, J., He, Z., Yan, A., Chen, X., & McAuley, J. (2024). Bridging language and items for retrieval and recommendation. *arXiv*. <https://doi.org/10.48550/arXiv.2403.03952>
- Jannach, D., Manzoor, A., Cai, W., & Chen, L. (2021). A survey on conversational recommender systems. *ACM Computing Surveys*, 54(5), Article 105. <https://doi.org/10.1145/3453154>
- Kulkarni, M., Kim, K., Garera, N., & Trivedi, A. (2023). Label efficient semi-supervised conversational intent classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Industry Track)* (pp. 96–102). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.acl-industry.11>
- Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., & Mars, J. (2019). An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference*

on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 1311–1316). Association for Computational Linguistics. <https://doi.org/10.18653/v1/D19-1131>

Mohamed, H., Badawy, O., Eskaf, K., & Zaghloul, M. S. (2020). A virtual assistant chatbot for college advising system. In S. Smys, T. Senjyu, & P. Lafata (Eds.), *Second International Conference on Computer Networks and Communication Technologies (Lecture Notes on Data Engineering and Communications Technologies, Vol. 44, pp. 625–633)*. Springer, Cham. https://doi.org/10.1007/978-3-030-37051-0_70

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv. <https://doi.org/10.48550/arXiv.1910.01108>

Appendix A: Intent Labels and Operational Meanings

This appendix lists the twenty intent labels implemented by the classifier and the operational meaning used by the routing layer. The labels define the bounded functional scope of the evaluated chatbot.

Table (A.1): Intent labels and operational meanings

Intent Label	Operational Meaning
account management	Password, profile, and account-support requests.
add to cart	Add a specified product to the prototype cart.
billing and payment	Payment-method and billing questions.
compare products	Compare two selected or referenced products.
customer care support	General store-support assistance.
delivery options	Available or simulated delivery methods and times.
issue support	Support for damaged, incorrect, or problematic products.
order management	Manage or inspect existing orders.
order tracking	Track an order reference.
out of scope	Requests outside shopping, products, orders, delivery, payment, and support.
place order	Checkout or order-placement requests.
price filter	Find products under an explicit or inferred price limit.
product details	Retrieve detailed information for a selected product.
product inquiry	Ask whether a product or product type is available.
product recommendation	Request suitable or highly rated products.
product search	Search for products by product, brand, or title terms.
remove from cart	Remove an item from the prototype cart.
shipping cost	Ask about shipping fees.
store information	Ask about the store or its general policies.
submit feedback	Submit or discuss feedback about the store experience.

Appendix B: Executed Chatbot Scenario Matrix

The following matrix records the selected end-to-end chatbot scenarios used in the thesis. Raw intent and final route are retained separately so that deterministic corrections are not reported as additional classifier accuracy.

Table (B.1): Executed chatbot scenarios

ID	User Request	Raw Intent -> Final Route	Confidence	Observed Output
CT-01	Waterproof mascara under \$20	price_filter	98.6%	6 product cards
CT-02	Niacinamide serum for dry skin	product_search	47.2%	6 product cards
CT-03	Compare two eyeliners	compare_products	98.8%	2 product cards
CT-04	Shipping cost	shipping_cost	99.2%	Policy/support response
CT-05	Track ORD-123456	order_tracking	99.1%	Policy/support response
CT-06	Damaged product support	issue_support	99.1%	Policy/support response
CT-07	Visa payment	billing_and_payment	94.4%	Policy/support response
CT-08	Football joke	out_of_scope	99.5%	Policy/support response
CT-09	Misspelled mascara under \$15	remove_from_cart -> price_filter	80.5%	6 product cards
CT-10	Oral-B toothbrushes	product_inquiry	90.7%	6 product cards
CT-11	Checkout request	billing_and_payment -> place_order	97.7%	Policy/support response
CT-12	Password reset	account_management	99.2%	Policy/support response

Appendix C: Final Runtime Validation

The final focused runtime validation contained 15 checks and all 15 passed. The suite verifies service health, the 8,000-record catalog, product categories, availability of both trained models, price-constrained recommendation, comparison uniqueness, typographical-error recovery, shipping behavior, negative-budget handling, vague-request clarification, out-of-catalog and out-of-scope handling, blank-message rejection, unsupported model-key rejection, and overlong-message rejection. It complements, rather than replaces, the 52-case chatbot regression report.

Table (C.1): Final runtime, API, asset, and hidden-issue validation results

#	Verification Check	Result	Evidence
1	Health endpoint and application version	Passed	HTTP 200; status ok; version 1.1.0
2	Catalog contains 8,000 products	Passed	SQLite returned 8,000 product records
3	Catalog categories are available	Passed	Category endpoint returned beauty-product groups
4	DistilBERT and BERT-Tiny load locally	Passed	Both model packages returned valid predictions
5	Price-constrained mascara recommendation	Passed	Returned mascara products within the requested budget
6	Product comparison uses unique visible titles	Passed	Three distinct comparison items were rendered
7	Typographical-error recovery	Passed	Misspelled request was normalized and routed correctly
8	Shipping-policy request	Passed	Expected policy response returned
9	Negative budget is rejected	Passed	Controlled validation response; no products returned
10	Vague request asks for clarification	Passed	System requested missing product or budget details
11	Out-of-catalog product request	Passed	No unrelated beauty-product fallback was returned
12	Out-of-scope request	Passed	Bounded-domain response returned
13	Blank chat message is rejected	Passed	HTTP 422 validation response
14	Unsupported model key is rejected	Passed	HTTP 422 validation response
15	Overlong chat message is rejected	Passed	HTTP 422 validation response

The detailed results are retained in the project audit reports and the separate CHATBOT_FULL_TEST_REPORT.json and CHATBOT_FULL_TEST_REPORT.md files. The six final interface screenshots embedded in Figures (15)–(20) provide visual evidence for the approved identity, successful recommendation and comparison behavior, typographical-error recovery, and controlled validation states.

Appendix D: Installation and Execution Guide

The following steps reproduce the evaluated local execution. Paths may be adjusted for Windows, Linux, or macOS, but the model folders and SQLite database must remain in the locations expected by the backend configuration.

D.1 Backend

```
python -m venv .venv
# Windows
.venv\Scripts\activate
# Linux/macOS
source .venv/bin/activate
pip install -r requirements.txt
python backend/run_server.py
```

The backend initializes the SQLite schema before importing services. The default API is `http://127.0.0.1:8000`, and `GET /api/health` should return status ok and version 1.1.0. Transformers is optional at runtime because the packaged DistilBERT and BERT-Tiny artifacts can be loaded by the project-local PyTorch and safetensors implementation.

D.2 Frontend

```
cd frontend
npm install
npm run dev
```

For source development, Vite serves the interface at `http://127.0.0.1:5173`. The supplied `frontend/dist` directory is the tested prebuilt interface and can be served with any local static server. The hero and product images are local. `VITE_API_BASE_URL` may be used when the backend runs at a different address.

D.3 Verification

```
python backend/scripts/validate_chatbot_expected_results.py
python tests/runtime_validation.py
```

The first command reproduces the 52-case chatbot regression report. The second reproduces the 30-check runtime validation. Review the two Markdown reports, JSON files, final execution logs, `FINAL_RUNTIME_AUDIT_AR.md`, and the screenshot folders before Moodle submission.

Appendix E: Screenshot Evidence Manifest

Sixteen screenshots were retained as visual evidence. Success images show correct execution, while error/recovery images show controlled responses to invalid input, absent context, unsupported requests, and backend unavailability.

Table (E.1): Screenshot evidence files

Category	File	Evidence
Success	success/01_homepage_approved_identity.png	Approved homepage identity and local hero asset.
Success	success/04_chat_price_recommendations.png	DistilBERT price-constrained recommendations.
Success	success/05_chat_comparison.png	Comparison of distinct product results.
Success	success/06_chat_typo_recovery.png	Recovery from a misspelled user request.
Validation	errors/02_negative_budget_validation.png	Controlled rejection of a negative budget.
Validation	errors/04_out_of_catalog_scope.png	Controlled response to a product outside the catalog.

Appendix F: Moodle Submission Checklist

- ☐ Confirm that the college-approved Word template has been applied to the cover, margins, headers, footers, and signature pages.
- ☐ Update researcher e-mails, ORCID identifiers, signatures, and any university-required metadata.
- ☐ Update the table of contents, list of figures, and list of tables in Microsoft Word.
- ☐ Run final spelling and grammar review in English and Arabic.
- ☐ Confirm that every figure and screenshot is readable in the final PDF.
- ☐ Package the thesis, presentation, corrected source code, trained models, database, execution instructions, and verification evidence for Moodle.
- ☐ Run the institutionally approved similarity and AI-use checks. Revise highlighted wording in the researchers' own academic voice and retain accurate citations.
- ☐ Confirm that all supervisor comments are closed and that submission occurs within the official deadline.