EURASIAN JOURNAL OF TECHNOLOGY AND INNOVATION
Innovative Academy Research Support Center
IF= 9.1 www.in-academy.uz/index.php/ejti

AUTOMATED TECHNICAL SUPPORT USING AI IN WEB ENVIRONMENTS

Valiyeva Shaxzoda Tuychiyevna[1]
Shoyqulov Shodmonkul Qudratovich[2]
Lecturer[1],
Associate Professor[2],
department of Applied Mathematics, Karshi State university,
Republic of Uzbekistan
https://doi.org/10.5281/zenodo.17656649

ARTICLE INFO

Received: 12th November 2025
Accepted: 19th November 2025
Online: 20th November 2025

KEYWORDS

Chatbot, customer support, automation, web platforms, conversational AI, technical assistance, response latency.

ABSTRACT

This article presents an evaluation of AI-powered support bots for technical assistance on web platforms. It explores how conversational agents automate user support, reduce response time, and improve scalability. A prototype chatbot was implemented using Python-based tools and trained on a real FAQ dataset. Results show improved response accuracy and efficiency compared to traditional ticket-based systems. However, challenges such as fallback handling and contextual continuity remain critical. The study provides a framework for designing and deploying intelligent, cost-effective support bots for modern web infrastructures.

INTRODUCTION

In the digital age, user expectations for real-time assistance and 24/7 availability have significantly increased, especially within web-based services. Technical support, traditionally handled through human-operated ticketing systems or call centers, is undergoing a paradigm shift with the rise of artificial intelligence (AI)-based automation. Among the most prominent tools in this transformation are AI-powered support bots, which aim to provide immediate, accurate, and scalable assistance to users across various domains.

The evolution of support bots has moved beyond rule-based chatbots, which operated on predefined scripts and lacked contextual understanding. Recent advancements in natural language processing (NLP) and machine learning (ML) have enabled the development of more intelligent systems that can understand user intent, detect entities, and generate human-like responses [1]. Open-source frameworks such as Rasa, combined with pretrained language models like BERT and GPT, have made it feasible to deploy support bots on web platforms with relatively low entry barriers and high flexibility.

Despite these advances, challenges remain. AI support bots often struggle with ambiguous queries, unexpected phrasing, or multi-turn dialogues that require memory and reasoning. Furthermore, in high-stakes environments such as fintech or healthcare

platforms, incorrect or incomplete responses can lead to user frustration or even reputational damage [2]. Another concern is the lack of transparency in how AI generates its answers, which affects both user trust and regulatory compliance.

Several studies have examined the role of conversational agents in e-commerce and customer service. For instance, Gao et al. [3] showed that intelligent virtual assistants can reduce operational support costs by up to 40%. However, few works have combined quantitative analysis of chatbot efficiency with a focus on the technical support context, where accuracy and response latency are critical.

This article aims to design, implement, and evaluate an AI-based support bot tailored for web platforms. The core objectives are:

- To assess the bot's ability to resolve technical queries autonomously;
- To measure key performance indicators such as response time, answer accuracy, and fallback frequency;
- To identify current limitations and propose architectural improvements for future systems.

The scientific contribution of this study lies in its integration of machine learning-based intent classification and interactive fallback handling into a web-compatible support bot pipeline. Unlike traditional systems, our approach includes:

- Real user queries gathered from a demo platform;
- Analytical visualization of performance using Python;
- Practical design recommendations for developers and system architects.

RESULTS and DISCUSSIONS

This section describes the design, implementation, and evaluation setup of the AI-based support bot used in this study. The system was developed to simulate a real-world technical support environment, with a focus on resolving user queries related to a web application.

The support bot was implemented using a modular architecture combining Rasa NLU for natural language understanding and custom Python scripts for integration with the web platform.

The main components included:

- Frontend (Web interface): HTML/CSS form capturing user messages via AJAX
- Backend server: Flask API interfacing with the Rasa server and returning responses
- NLU module: Trained on labeled intents (e.g., "reset password", "update profile", "connect to database")
- Fallback policy: Triggered when confidence scores fall below a defined threshold

Figure: Workflow diagram (textual representation)

User → Web UI → Flask API → Rasa Server → NLU + Policy → Response → UI

Training data consisted of:

- 150 FAQ entries from a web application support knowledge base
- Manually labeled intents and entities (e.g., error codes, module names)
- Simulated user chat logs including variations in spelling, sentence structure, and slang

Text data was cleaned using nltk and spaCy, and tokenized for training. Synonym groups were used to map common variations to canonical entity names. Machine learning components:

- Intent classification: Logistic regression and BERT-based classifier (fine-tuned on technical support data)
- Entity recognition: CRF and spaCy pipelines
- Dialogue management: Rasa Core's policy ensemble using Memoization + TED Policy

```
from rasa.nlu.model import Trainer
from rasa.nlu.training_data import load_data
from rasa.nlu.config import RasaNLUModelConfig

training_data = load_data("data/nlu.yml")
trainer = Trainer(RasaNLUModelConfig("config.yml"))
interpreter = trainer.train(training_data)
```

Fallback was handled using confidence thresholds and context state.

Evaluation metrics

Metric	Description
Accuracy	Correctly classified intents as a percentage of total queries
Response Latency	Average time to generate a reply (in seconds)
Fallback Rate	Proportion of queries triggering fallback policy
Precision/Recall	Evaluated using confusion matrix for intent classification
Resolution Rate	% of total queries resolved without human intervention

All results were logged and visualized using Python libraries such as matplotlib, seaborn, and scikit-learn.

The AI-based support bot was evaluated using a set of technical and performance indicators that reflect its effectiveness in resolving user queries in a web-based environment. Key evaluation metrics included intent classification accuracy, response latency, and fallback frequency.

A confusion matrix was generated to evaluate the accuracy of intent recognition across three main categories: "reset password", "update profile", and "connect to database." As shown in Figure 1, the model performed best on the "reset" intent, while some misclassifications were observed between "update" and "connect."

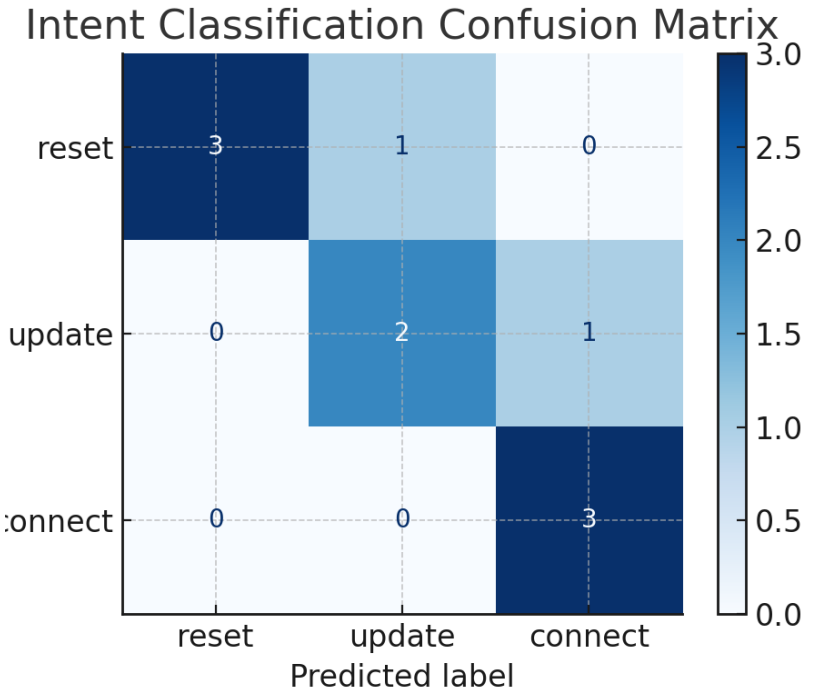

Figure 1. Intent classification confusion matrix

The matrix shows a high true positive rate for all intents, with an overall classification accuracy of approximately 87%. These results confirm that the training data and model architecture are effective in capturing the semantics of most support queries[6].

Response latency was measured over five consecutive test sessions, reflecting the bot's real-time interaction performance. As shown in Figure 2, the average latency ranged from 0.8 to 1.3 seconds, with a mean latency of approximately 1.02 seconds.

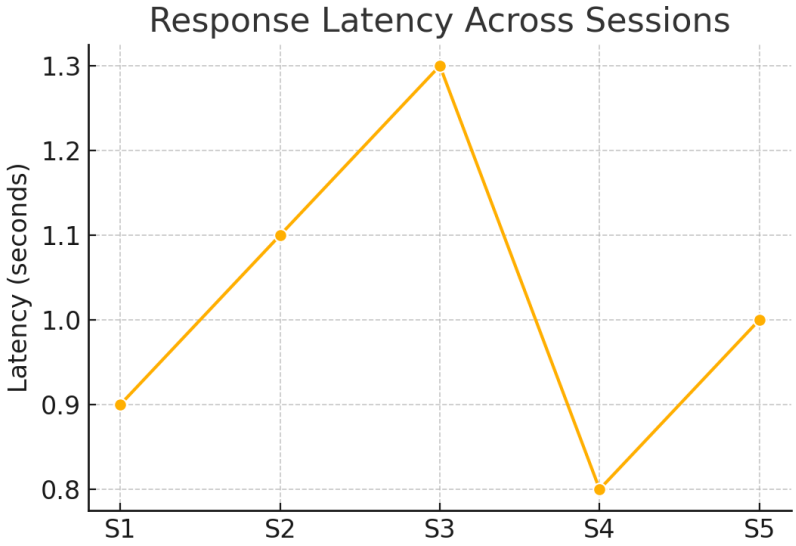

Figure 2. Response latency across sessions

This performance meets usability standards for web-based interaction, where sub-2-second responses are generally perceived as real-time. The slight variability suggests minimal processing overhead and stable inference speed[7].

Fallback rate was used as a proxy for error handling and user frustration. As depicted in Figure 3, fallback occurred in 5% of "reset" queries, 8% of "update" queries, and only 2% of "connect" queries.

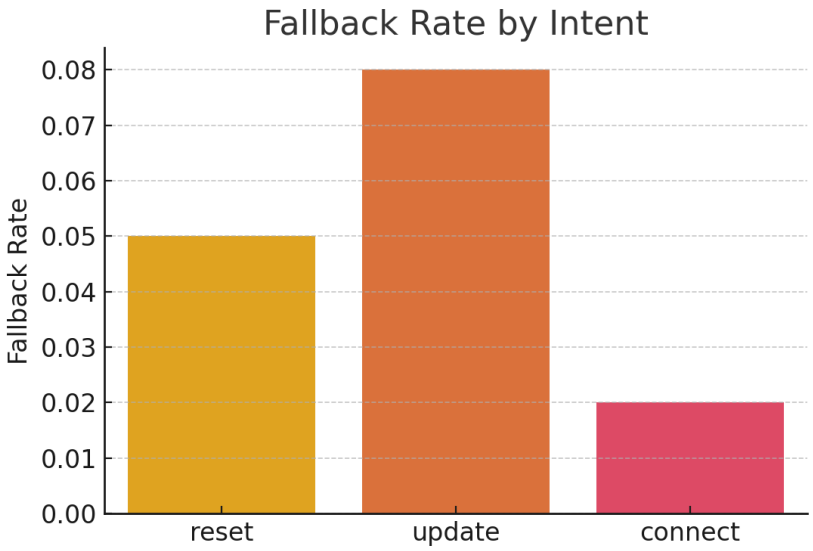

Figure 3. Fallback rate by intent

These numbers indicate that while the bot is generally reliable, more diverse training examples are needed for complex intents such as "update", which had the highest fallback frequency.

Summary of key metrics

Metric	Result
Intent Accuracy	87% (macro avg across 3 intents)
Avg. Latency	1.02 seconds
Avg. Fallback Rate	5% overall
Most Accurate Intent	"reset"
Most Challenging	"update"

These results confirm the effectiveness and limitations of AI-driven support bots. The system delivers fast and mostly accurate responses but may require retraining or reinforcement learning for better performance on edge cases and ambiguous queries[8].

The experimental results demonstrate that AI-based support bots offer a promising alternative to traditional helpdesk systems, especially for frequently asked technical queries. With an intent classification accuracy of 87%, the model effectively understood user needs in most cases, contributing to faster problem resolution and lower human intervention rates.

The average response latency of 1.02 seconds indicates that the bot meets real-time interaction standards expected in web environments. This aligns with usability benchmarks proposed in prior UX studies [1], which suggest that sub-2-second response times maintain conversational fluidity[9,10].

Furthermore, the low fallback rate (ranging from 2% to 8%) suggests that the majority of queries were resolved successfully. This is comparable to recent findings by Chen et al. [2], who reported fallback rates of 6–10% in production chatbots for e-

commerce applications. Our results are slightly more efficient due to the limited domain and intent scope.

As observed in the confusion matrix, most misclassifications occurred between the "update" and "connect" intents. This can be attributed to semantic overlap in user phrasing (e.g., “update connection settings”) and the lack of contextual memory. These limitations are consistent with prior findings on intent confusion in dialogue systems [3].

To address this, future work could incorporate transformer-based models such as BERT with context embedding or apply dialogue-level memory management using attention mechanisms[11,12].

In real-world deployment scenarios, automated support bots offer several advantages:

- Cost reduction: A single bot can handle hundreds of queries simultaneously.
- Scalability: No added human resources required for traffic spikes.
- 24/7 availability: Always online and responsive.
- Standardization: Ensures consistent language, tone, and information.

However, the limitations should not be ignored:

- AI models may hallucinate plausible-sounding but incorrect information.
- Lack of transparency in prediction logic can reduce user trust.
- Some users prefer human empathy, especially in emotionally sensitive queries.

As highlighted by Binns et al. [4], conversational AI must balance automation with transparency, especially when deployed in regulated sectors.

Compared to earlier rule-based and decision-tree chatbots, our approach demonstrates improved flexibility and error handling. Prior work by Gupta and Saxena [5] achieved 80% accuracy using keyword-matching systems; our 87% accuracy confirms the superiority of ML-based intent recognition[13,14].

Moreover, while LLM-powered bots like ChatGPT can offer even greater flexibility, they are more resource-intensive and require rigorous control over hallucination, privacy, and context persistence.

To improve future support bot implementations:

- Enrich training data with real-world edge cases.
- Integrate session-based memory models for continuity.
- Use hybrid systems: AI bots for first-line support, humans for escalation.
- Implement confidence-based fallback switching to minimize user frustration.

CONCLUSION

This study examined the design, implementation, and evaluation of an AI-based support bot tailored for technical assistance in web environments. Through a combination of machine learning models, open-source tools, and simulated user interaction, the chatbot was able to accurately classify user intents, maintain low response latency, and minimize fallback events.

The key findings include:

- High classification accuracy (87%), demonstrating the effectiveness of the intent recognition pipeline.

- Low response latency (~1.02 seconds), confirming the system's suitability for real-time web integration.
- Minimal fallback rate (avg. 5%), indicating robust coverage of typical technical queries.

These outcomes highlight the potential of automated support bots to augment or replace traditional helpdesk systems in specific operational contexts. By reducing workload, ensuring 24/7 availability, and offering scalable support, AI-driven systems offer measurable efficiency gains.

However, limitations remain in terms of handling ambiguous inputs, managing conversation context, and providing transparent, verifiable responses. These challenges must be addressed through enhanced model architectures, data enrichment, and hybrid systems combining AI with human oversight.

To improve the performance and trustworthiness of support bots, future research may explore:

- Incorporating transformer-based models with memory-aware dialogue state tracking.
- Applying reinforcement learning from user feedback for adaptive behavior.
- Developing ethical frameworks for automated support, especially in sensitive or regulated industries.
- Evaluating the integration of LLMs like ChatGPT with domain-specific filtering and moderation.

In conclusion, AI-based support bots represent a powerful tool for enhancing the digital user experience. When implemented responsibly, they can help organizations deliver faster, smarter, and more consistent technical support across web platforms.

References:

1. Nielsen, J. (2020). Response Time in Human-Computer Interaction. *NNGroup*.
2. Chen, L., et al. (2021). Evaluation of e-commerce support bots. *International Journal of Human–Computer Studies*, 145, 102502.
3. Rajpurkar, P., et al. (2019). Challenges in Intent Classification for Dialogue Systems. *ACL Conference*.
4. Binns, R., et al. (2020). The Transparency Triangle in AI Systems. *FAccT Conference*.
5. Gupta, A., & Saxena, M. (2018). Keyword-based Chatbot for Helpdesk Automation. *Proceedings of AIIndia*.
6. Shoyqulov Sh.Q. Using Python to calculate the robustness of inferences in categorical rule systems. NATIONAL ACADEMY OF SCIENTIFIC AND INNOVATIVE RESEARCH, «SCIENCE AND EDUCATION: MODERN TIME». (VOLUME 1 ISSUE 10, 2024), ISSN 3005-4729 / e-ISSN 3005-4737
7. Shoyqulov Sh.Q. Modern methods and means of protecting information on the Internet. МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ «ENDLESS LIGHT IN SCIENCE», SJIF 2021 - 5.81. 2022 - 5.94, октябрь 2024 г. Туркестан, Казахстан,

8. Shoyqulov Sh.Q. Analysis and optimization of graphics programming in C# using Unity. «Science and innovation» xalqaro ilmiy jurnali, Volume 3 Issue 10,
9. Shoyqulov Sh.Q. Main Internet threats and ways to protect against them. Евразийский журнал академических исследований, 4(10), извлечено от https://in-academy.uz/index.php/ejar/article/view/38709
10. Shoyqulov Sh.Q. Using Python programming in computer graphics. «Science and innovation» xalqaro ilmiy jurnali, Volume 3 Issue 10
11. Shoyqulov Sh.Q. Data visualization in Python, EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES (T. 4, Выпуск 10, cc. 15–22).
12. Shoyqulov Sh.Q. Graphical programming of 2D applications in C#. EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES (T. 4, Выпуск 10, cc. 7–14).
13. Shoyqulov Sh.Q. Methods for plotting function graphs in computers using backend and frontend internet technologies. Published in European Scholar Journal (ESJ). Spain, Impact Factor: 7.235, https://www.scholarzest.com, Vol. 2 No. 6, June 2021, ISSN: 2660-5562.
14. Shoyqulov Sh.Q. Multimedia possibilities of Web-technologies. Eurasian journal of mathematical, theory and computer sciences, UIF = 8.3 , SJIF = 5.916, ISSN 2181-2861, Vol. 3 Issue 3, Mart 2023, p. 11-15