УДК 004.89:004.415.2

ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ОБЛІКУ ЗВЕРНЕНЬ ДО СЛУЖБИ ПІДТРИМКИ З ВИКОРИСТАННЯМ FIREBASE

Судак В.В[1]; Крісілов В. А., д.т.н., професор[2]
[1] Національний університет «Одеська політехніка», Навчально-науковий інститут комп'ютерних систем
[1] Україна, м. Одеса
[1] sudak.v.vova@gmail.com; [2] krisilov@op.edu.ua

Анотація. *Проєктування та розроблення вебзастосунку для обліку звернень до служби підтримки. Запропоновано архітектуру на основі React, Firebase і серверних Cloud Functions на Node.js, що інтегрують мовну модель для визначення категорії, пріоритету звернень та чернеток відповідей. Результати ШІ мають рекомендаційний характер і підтверджуються оператором. Описано методику тестування.*
Ключові слова: *служба підтримки; облік звернень; Firebase; React; Cloud Functions; штучний інтелект; тестування.*

Вступ. Служби технічної підтримки опрацьовують запити, що надходять різними каналами, тому за відсутності централізованого обліку зростає ризик втрати звернень, затримок відповіді та нерівномірного навантаження на операторів. Комерційні платформи (Zendesk, Freshdesk, Jira Service Management) реалізують базовий цикл опрацювання звернень, проте їхні інтелектуальні можливості залежать від тарифу й конфігурації, а інтеграція з власною хмарною інфраструктурою обмежена. Водночас сучасні дослідження демонструють можливість застосування великих мовних моделей для автоматичної класифікації звернень до служб підтримки, хоча якість результатів залежить від обраної моделі та способу її налаштування [1]. Хмарна платформа Firebase [2] надає готові сервіси автентифікації, документо-орієнтованого сховища та хостингу, що скорочує час створення працездатного прототипу.

Мета дослідження. Підвищення точності класифікації та опрацювання звернень шляхом проєктування та розроблення вебзастосунку SupportTech з інтелектуальним компонентом, який визначає категорію й пріоритет звернення та формує чернетку відповіді для оператора, а також розроблення методики перевірки коректності його роботи.

Основна частина. Проєктні рішення. Система має три ролі: клієнт створює звернення й відстежує його статус; оператор працює з чергою, коригує результати аналізу та надсилає відповідь; адміністратор керує обліковими записами й ролями. Структуру даних адаптовано до моделі Cloud Firestore [2]: визначено сім колекцій і підколекцій (users, categories, tickets, messages, events, notifications, responseTemplates). Листування та історію змін винесено в підколекції звернення через обмеження розміру документа в 1 МіБ, вкладення (до 10 МБ) зберігаються в Cloud Storage, а у Firestore – лише їхні метадані. Результати інтелектуального аналізу записуються в окреме поле aiAnalysis (стан, запропоновані категорія й пріоритет, чернетка), відокремлене від значень, підтверджених оператором.

Програмна реалізація. Клієнтську частину виконано як односторінковий застосунок на React [3] із TypeScript, зібраний інструментом Vite; маршрутизацію забезпечує React Router, адаптивний інтерфейс – бібліотека MUI. Шари, визначені під час проєктування, відображено в структурі коду: сторінки й хуки відповідають контролерам, а модулі TicketService,

MessageService, AuthService інкапсулюють доступ до Firebase SDK. Зміни статусів і нові повідомлення відображаються без перезавантаження сторінки завдяки real-time слухачам onSnapshot. Серверну частину реалізовано як Cloud Functions на Node.js: тригер onTicketCreated записує подію в історію, створює сповіщення й запускає аналіз, onStatusChanged фіксує зміну статусу та час першої відповіді, а HTTPS-функція generateDraft формує чернетку за запитом оператора. Під час аналізу функція передає мовній моделі лише тему й опис звернення та перелік допустимих категорій і вимагає відповідь у форматі JSON, яку перевіряє за схемою zod; некоректний результат відхиляється. Час очікування обмежено 20с (AbortController): у разі помилки, таймауту або вичерпання квоти аналіз отримує стан failed, а звернення залишається доступним для ручного опрацювання. Ключ API зберігається в Secret Manager і недоступний клієнтському коду. Розмежування прав забезпечують custom claims, що встановлюються через Admin SDK під час зміни ролі, та Firebase Security Rules: клієнт читає лише власні звернення, оператор – чергу, змінювати ролі може тільки адміністратор.

Тестування. Для модуля опрацювання звернення оператором застосовано метод функціональних діаграм причинно-наслідкових зв'язків і таблиць рішень [4]. Причинами обрано: автентифікацію оператора, доступність звернення, наявність результатів аналізу, коректність категорії та пріоритету після коригування, явне підтвердження відповіді й успішність запису в базу даних; наслідками – відображення рекомендацій, збереження відповіді зі зміною статусу, повідомлення про помилку та перехід до ручного опрацювання. Повну таблицю рішень скорочено до мінімального набору правил, кожне з яких перетворено на тестовий сценарій. Перевірку виконано на трьох рівнях: модульні тести сервісів і схеми валідації (Vitest, React Testing Library); тести Security Rules і Cloud Functions у Firebase Emulator Suite з пакетом @firebase/rules-unit-testing, вимога – 100 % успішних перевірок рольових обмежень. Оцінювання інтелектуального компонента на розміченому наборі не менше ніж зі 100 звернень, не використаних під час налаштування запиту, за метриками accuracy (поріг ≥ 80 %) та F1 для пріоритету (поріг ≥ 0,75). Окремо перевірено сценарії недоступності мовної моделі: звернення не втрачається й лишається в черзі. Результати: accuracy класифікації – 86 %, F1 пріоритету – 0,78, успішно пройдено 19 із 20 сценаріїв.

Висновки. Спроєктовано й реалізовано вебзастосунок SupportTech на базі React, Firebase та Cloud Functions на Node.js, що охоплює повний цикл опрацювання звернення – від створення клієнтом до відповіді оператора. Інтелектуальний компонент ізольовано від основного облікового процесу: його результати лише рекомендують значення, а помилки не блокують роботу системи. Запропонована методика тестування на основі причинно-наслідкових діаграм, перевірки правил безпеки та оцінювання якості класифікації дає змогу об'єктивно підтвердити відповідність вимогам.

Використання інструментів ШІ. Claude Sonnet 5.5 (Anthropic) використано: 1) для перекладу англійською назви, анотації та ключових слів; 2) для перевірки пунктуації й орфографії тексту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Koukal, F., Dařena, F., Ježdík, R., & Přichystal, J. (2024). Improving automated categorization of customer requests with recent advances in natural language processing. European Journal of Business Science and Technology, 10(2), 173–184. https://doi.org/10.11118/ejobsat.2024.010

[2]Google. (n.d.). Firebase documentation: Cloud Firestore, Authentication, Cloud Functions, Security Rules. Retrieved November 5, 2026, from https://firebase.google.com/docs

[3]Meta Open Source. (n.d.). React documentation. Retrieved November 5, 2026, from https://react.dev
[4] Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing (3rd ed.). John Wiley & Sons.

Sudak V.V.; Krisilov V. A.

Design and development of a system for tracking support tickets using Firebase

Abstract. *Design and development of a web application for tracking support service requests. An architecture based on React, Firebase, and server-side Cloud Functions on Node.js is proposed, which integrates a language model to determine the category and priority of requests and to generate draft responses. AI results are recommendations and must be verified by an operator. The testing methodology is described.*

Keywords: *support service; ticket tracking; Firebase; React; Cloud Functions; artificial intelligence; testing.*