https://doi.org/10.31891/2219-9365-2026-87-19
UDC 004.89:004.451

LUKIANCHUK Iurii
Lutsk national technical university
https://orcid.org/0000-0001-9690-6197
e-mail: iuriilukianchuk87@gmail.com
BOYKO Lev
Lutsk national technical university
https://orcid.org/0009-0001-0117-8551
e-mail: l.boyko@lntu.edu.ua

DEVELOPMENT OF A CHATBOT FOR TRAFFIC ARBITRATION WITH AN INTEGRATED REWARD SYSTEM BASED ON DIGITAL COINS

This paper presents the results of developing a specialised chatbot aimed at optimising daily operations within traffic arbitration teams. Traffic arbitration teams face numerous challenges, including communication difficulties, excessive workload from routine tasks, unstructured workflows, and the lack of convenient progress tracking tools. The proposed software solution addresses these challenges by incorporating a staff motivation mechanism based on an internal virtual currency – coins, combining task distribution, progress tracking, analytics integration, and motivational incentives in a single Telegram-based interface. The chatbot architecture follows asynchronous programming principles using the Aiogram framework, which supports finite-state machine models essential for multi-step user dialogues. The document-oriented MongoDB database management system was chosen for its flexibility in storing user profiles, task histories, coin transactions, and referral data. The server-side component is deployed on cloud platforms such as Render or Railway, providing automatic updates and secure webhook handling. The study provides functional and structural models of the system, including UML class and component diagrams, as well as flowcharts of the main algorithms. A comparative analysis of existing similar solutions, such as ZeyBot, Everbot, Standuply, and Karma Bot, is performed, justifying the choice of the technology stack. The testing results confirm the stable operation of the developed application under load conditions close to real-world usage scenarios, with an average response time of less than half a second under a load of one hundred simultaneous users. Special attention is paid to information security issues, including HTTPS encryption, Telegram ID-based authentication, database access restrictions, and logging of administrative actions. The prospects of integrating artificial intelligence elements, such as intelligent traffic management and natural language processing, are also discussed.

Keywords: traffic arbitration, chatbot, task automation, reward system, digital coin, MongoDB, Aiogram.

ЛУК'ЯНЧУК Юрій, БОЙКО Лев
Луцький національний технічний університет

РОЗРОБКА ЧАТ-БОТА ДЛЯ АРБІТРАЖУ ТРАФІКУ З ІНТЕГРОВАНОЮ СИСТЕМОЮ ЗАОХОЧЕНЬ У ВИГЛЯДІ ЦИФРОВИХ КОЇНІВ

У статті висвітлено результати створення спеціалізованого чат-бота, призначеного для оптимізації щоденних операцій у командах, що працюють у сфері арбітражу трафіку. До складу програмного рішення вбудовано механізм мотивації персоналу на основі внутрішньої віртуальної валюти – коінів. Архітектура розробленого чат-бота ґрунтується на засадах асинхронного програмування з використанням фреймворку Aiogram та документо-орієнтованої системи керування базами даних MongoDB. У роботі наведено функціональні та структурні моделі системи, зокрема UML-діаграми класів і компонентів, а також блок-схеми основних алгоритмів. Виконано порівняльний аналіз наявних аналогічних рішень, таких як ZeyBot, Everbot, Standuply та Karma Bot, на основі чого обґрунтовано вибір технологічного стеку. Результати проведеного тестування засвідчили стабільну роботу розробленого застосунку в умовах навантаження, наближених до реальних сценаріїв використання. Окрему увагу приділено питанням захисту інформації та перспективам інтеграції елементів штучного інтелекту для підвищення ефективності системи.

Ключові слова: арбітраж трафіку, чат-бот, автоматизація завдань, система винагород, цифровий коїн, MongoDB, Aiogram.

Стаття надійшла до редакції / Received 20.04.2026
Прийнята до друку / Accepted 14.06.2026
Опубліковано / Published 10.09.2026

GENERAL STATEMENT OF THE PROBLEM AND ITS CONNECTION WITH IMPORTANT SCIENTIFIC OR PRACTICAL TASKS

The rapid evolution of digital marketing and affiliate programmes has led to a significant increase in the volume of internet traffic circulating through partner networks. Traffic arbitration, which involves purchasing low-cost advertising traffic and subsequently monetising it through affiliate programmes, has become a highly competitive field. Thousands of teams operate within this model, and each of them faces similar challenges on a daily basis: communication difficulties between team members, excessive workload on managers due to numerous routine tasks, unstructured workflows, and the absence of convenient tools for tracking progress and performance metrics. These issues create a substantial demand for automation solutions that can partially or fully delegate routine functions.

Chatbots have emerged as effective tools for automating internal business processes. According to a study conducted by Drift, more than fifty-five percent of companies worldwide have already integrated chatbots for customer service or internal task automation, and sixty-seven percent of users expect instant responses from digital services [1, 2]. Furthermore, gamification – the application of game mechanics in non-game environments – has been shown to increase employee engagement and productivity. A report by TalentLMS indicates that eighty-nine percent of employees feel more productive in a gamified environment [3]. In the context of traffic arbitration, gamification can take the form of an internal digital currency, commonly referred to as coins, which are awarded to team members for completing specific tasks. These coins can later be exchanged for bonuses, premium access to analytical tools, or other incentives defined by the team administrator.

However, most existing solutions are either too general and not adapted to the specifics of traffic arbitration, or they lack a flexible reward system that would motivate team members to work more efficiently. Therefore, the development of a specialised chatbot that combines task automation and an internal reward mechanism based on digital coins is a relevant scientific and practical task

ANALYSIS OF EXISTING RESEARCH AND PUBLICATIONS

Considerable research has been devoted to the automation of business processes using chatbots. The Drift study provides statistics on the implementation of chatbots across various economic sectors. The authors note that sixty-seven percent of users expect instant responses from digital services, which makes bot systems indispensable for rapid communication [2]. Gamification as a tool for increasing labour productivity is examined in the TalentLMS report, according to which eighty-nine percent of employees report that game mechanics make them more productive [3].

Existing software solutions for automating team processes include products such as ZeyBot from Zeydoo [4], Everbot for Slack [5], Standuply [6], and Karma Bot [7]. An analysis of these systems shows that none of them provide full integration of task automation, gamification, and specialisation for the arbitration model with an internal economy. For example, ZeyBot has integration with affiliate networks and a «karma coins» mechanism, but this development is proprietary, lacks open access, and has limited scalability. Everbot is deeply integrated with Slack and allows automation of recurring tasks, but it does not support Telegram or Discord and is not oriented towards arbitration. Standuply offers automated standups and API integrations, but it does not include a coin economy. Karma Bot implements a reward system and is multiplatform, but it lacks flexible task logic and connection to analytical data.

In works devoted to chatbot development, the Python programming language and the Aiogram framework are often used [8], providing high performance due to their asynchronous request processing model. MongoDB is increasingly chosen as the database for such systems because of its flexibility in working with JSON-like structures [9]. Thus, the analysis of literary sources confirms the relevance of developing a specialised chatbot for traffic arbitration with a reward system and allows justification of the choice of the technology stack

OBJECTIVES OF THE STUDY

The aim of this work is; to develop a chatbot for automating typical tasks in a traffic arbitration system with the implementation of a reward mechanism in the form of digital coins. To achieve this aim, the following objectives must be accomplished: to analyse the subject area of traffic arbitration teams and determine their main automation needs; to justify the choice of technologies, algorithms, and software tools for system implementation; to develop a functional-structural model of the chatbot, including algorithm flowcharts and UML diagrams; to design a database for storing information about users, tasks, coin transactions, and the referral system; to implement the chatbot code according to the developed architecture; to test the system under conditions close to real-world scenarios and evaluate its effectiveness; and to propose information security measures and prospects for integrating artificial intelligence elements

MAIN MATERIAL PRESENTATION

The chatbot development was carried out taking into account the requirements for performance, scalability, and reliability. The architecture is based on an event-driven principle, where each user request or action is treated as an event that activates a corresponding handler. This approach allows efficient servicing of thousands of simultaneous users. The Python programming language was chosen due to the availability of powerful libraries for chatbot development, particularly aiogram, support for asynchronous programming through asyncio, and a large community. The Aiogram framework provides convenient work with the finite-state machine model, which is necessary for implementing multi-step dialogues [8].

MongoDB was used as the database management system. Its document-oriented model is ideal for storing unstructured or semi-structured data, such as user profiles, task histories, and coin transactions. Using MongoDB allows easy scaling of the system and dynamic changes to the data schema without migrations [9]. The server-side part of the chatbot is deployed on cloud platforms such as Render or Railway, which support continuous updates from the Git repository, automatic logging, and webhook operation via HTTPS [10]. For integration with analytical systems, such as the Binom API, a separate module is provided that allows obtaining data on clicks, conversions, and expenses

in real time [11].

Several types of diagrams were developed to visualise the architecture. Figure 1 shows a functional-block diagram illustrating the main modules of the chatbot and the connections between them.

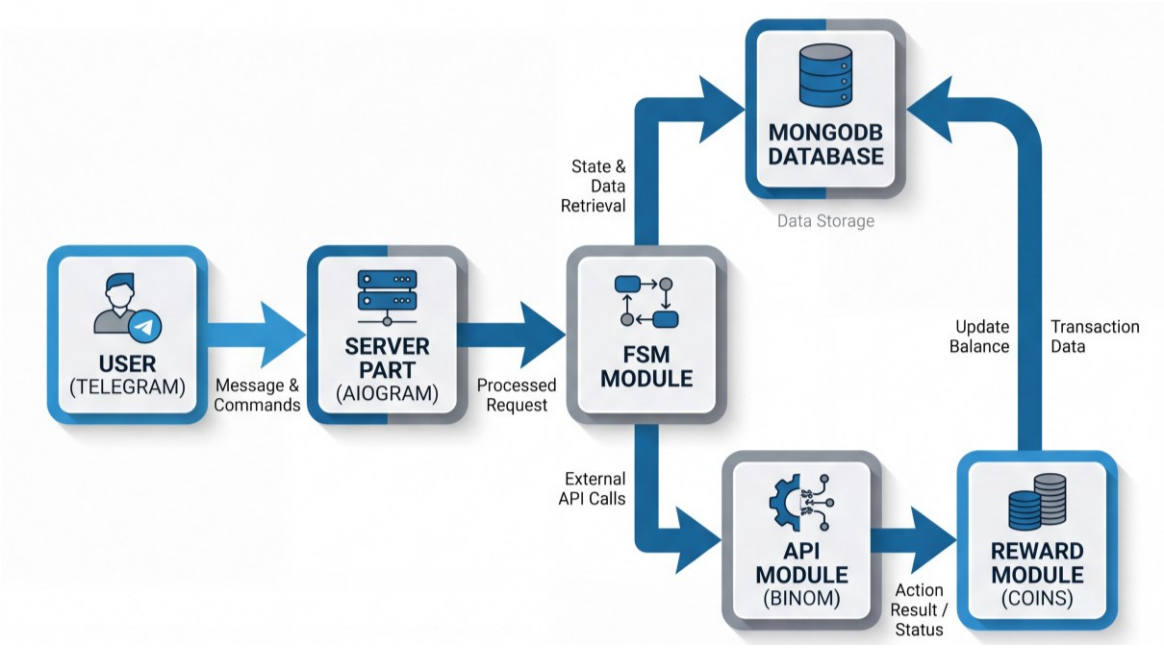

Fig. 1. Functional-block diagram of the chatbot modules

The component diagram (Figure 2) demonstrates the architectural structure of the system with distribution into separate functional blocks.

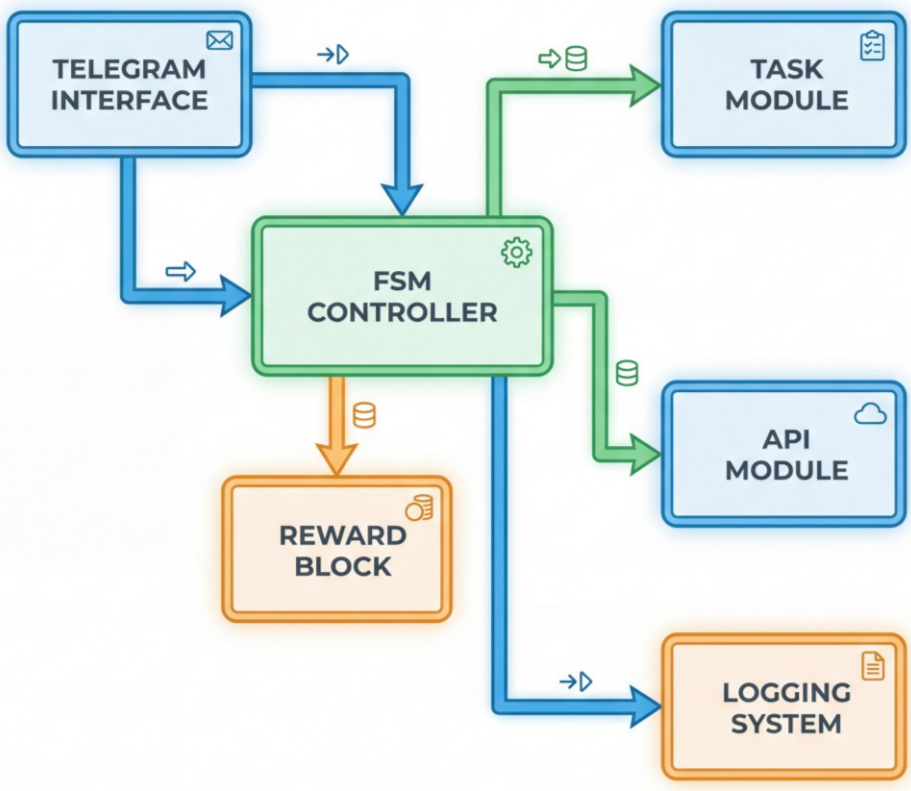

Fig. 2. Component diagram of the chatbot

The UML class diagram (Figure 3) displays the structure of the program code and the relationships between the main classes.

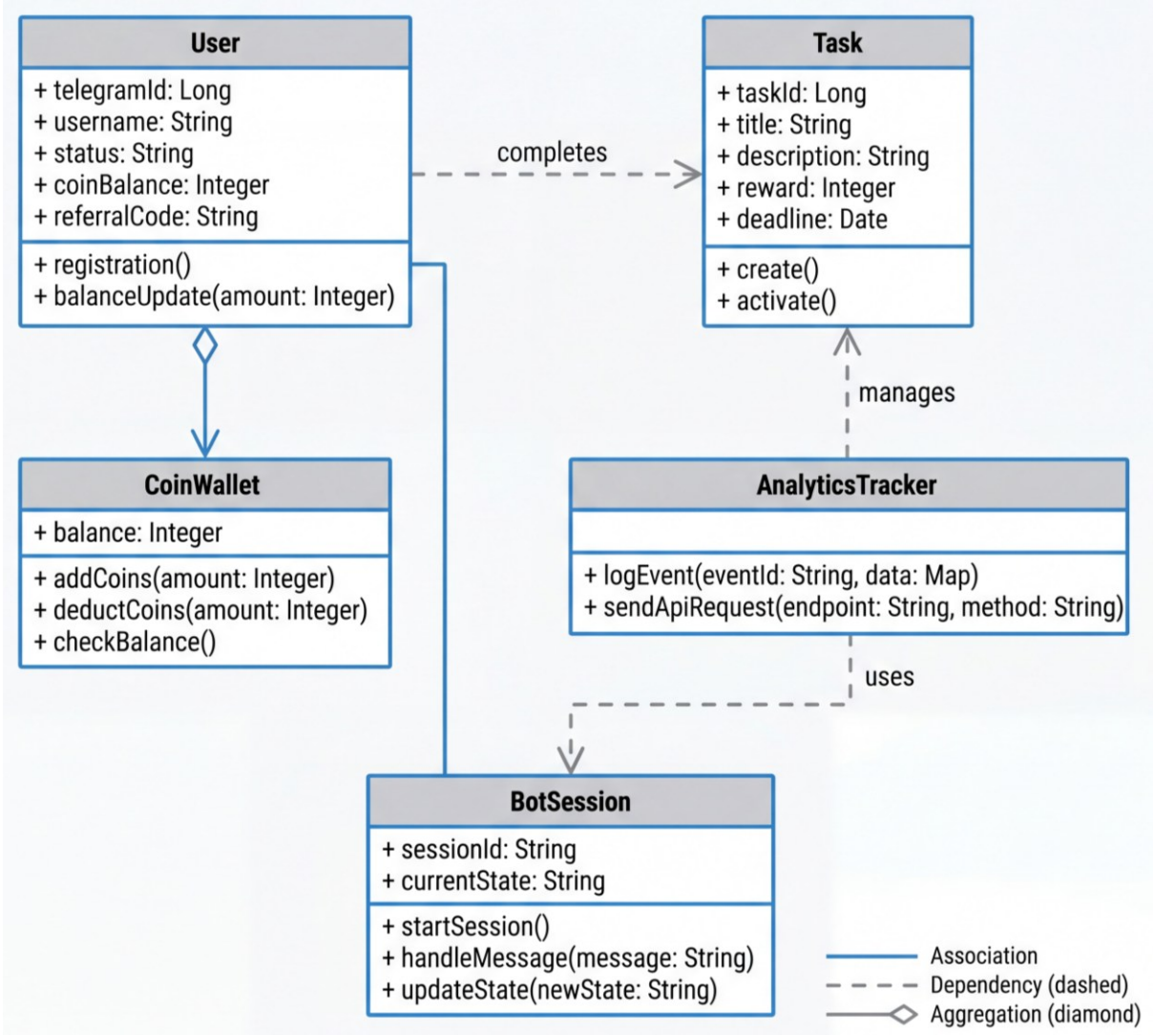

Fig. 3. Class diagram of the system

A MongoDB database structure was developed, consisting of six main collections. The «users» collection stores information about users: Telegram ID, username, status, coin balance, referral code, information about who invited the user, registration date, and task history. The «tasks» collection contains task descriptions: task ID, title, description, type, reward, deadline in days, creation date, and active flag. The «user_tasks» collection records task completion by specific users, storing user and task identifiers, status (in progress, completed, failed), start and completion times, proof link, and verification flag. The «coin_transactions» collection stores transaction history: user ID, amount, transaction type (reward, spend, referral bonus), description, and timestamp. The «sessions» collection is used for finite-state machine operation and stores Telegram ID, last action, current FSM state, and last update time. The «referrals» collection contains data about referral relationships: referrer ID, referred user ID, bonus amount, and timestamp.

The key element of the chatbot's logic is the finite-state machine model. Figure 4 presents a flowchart of the user interaction algorithm.

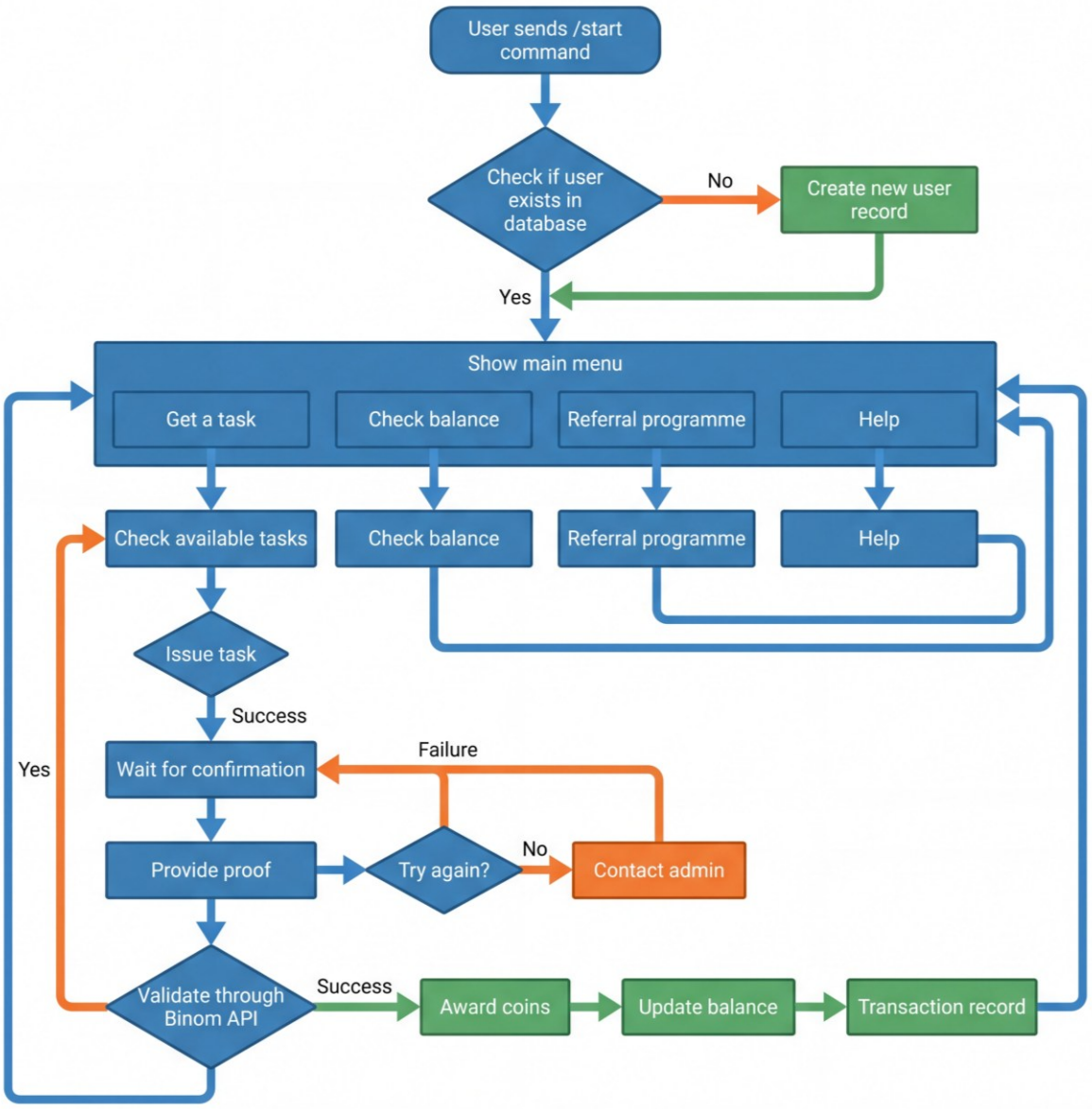

Fig. 4. Flowchart of user interaction algorithm

Figure 5 shows an activity diagram that illustrates the variability of user actions and possible paths for executing scenarios.

The chatbot implementation was carried out in Python using the aiogram library. The code is structured according to a modular principle: separate files for command handlers, FSM states, database operations, external API interactions, and configuration. Special attention was paid to security. All confidential data, such as bot tokens, MongoDB connection URIs, and API keys, are stored in environment variables and do not end up in the repository. The database connection is made via a secure channel using SSL certificates. The coin awarding mechanism is implemented as an atomic operation: first, the right to receive a reward is checked through the API or proof verification, then the user's balance is updated in MongoDB, and only after that a record is created in the transactions collection. In case of a failure at any stage, changes are rolled back, ensuring data consistency.

Three types of testing were conducted to verify system functionality: unit testing, integration testing, and manual transaction testing. Unit testing was performed using the Jest framework. Individual functions for user creation, input validation, and reward calculation were tested. Integration testing was conducted using the Supertest library. API routes were tested: user registration, task list retrieval, task submission for verification, and completion confirmation by the administrator. Manual transaction testing included checking scenarios for awarding coins for task completion, fund spending, and referral bonuses. In all cases, the system demonstrated correct operation. The test results showed that the chatbot can withstand loads of up to one hundred simultaneous users without a significant increase in response time, with the average command processing time being less than half a second.

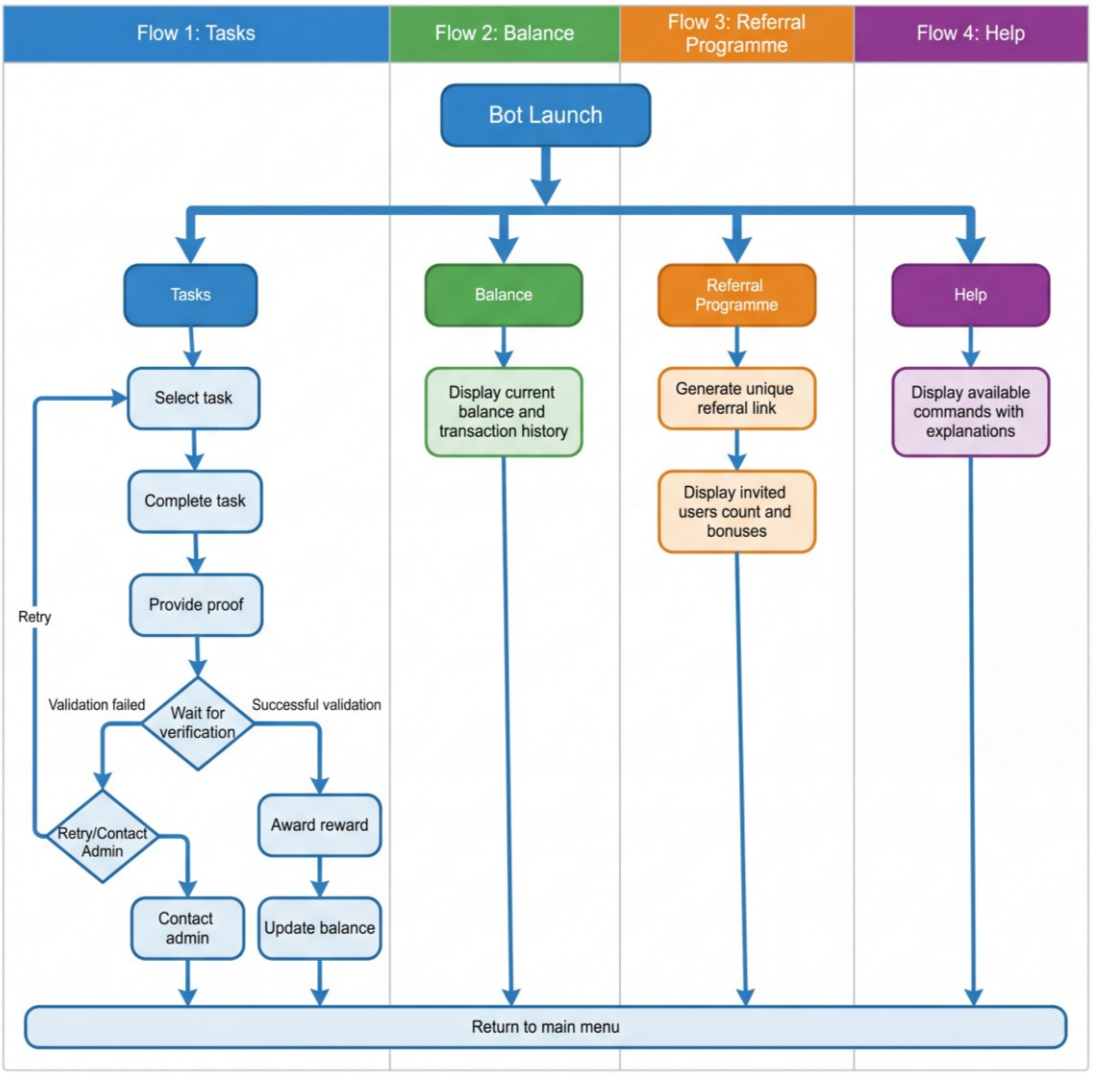

Fig. 5. User activity diagram

Given the sensitivity of the data being processed, a comprehensive set of security measures has been implemented in the system. The use of HTTPS with up-to-date SSL/TLS certificates ensures the protection of the communication channel between the client and the server. Authentication based on Telegram ID with additional verification through webhooks allows user identification. Database access restrictions are implemented by creating separate users for reading and writing. Regular database backups with copies stored in a secure environment ensure the possibility of data recovery in case of failure. Logging of all administrative actions and transactions allows auditing when necessary. Protection against SQL injection is ensured through the use of parameterised queries to MongoDB, and protection against DDoS attacks is provided through rate limiting.

Further system development envisages the integration of artificial intelligence elements. One possible direction is intelligent traffic management using classification algorithms such as Random Forest or CatBoost for automatic assessment of traffic source quality and forecasting of advertising campaign effectiveness. Another direction is the application of NLP models, particularly transformers like BERT or T5, for recognising user intentions and conducting natural dialogue instead of using button menus. The creation of recommendation systems that would suggest the most effective traffic sources to users based on analysis of their previous actions and the results of similar users is also promising. Additionally, anomaly detection methods such as Isolation Forest or autoencoders can be used to detect suspicious activity, such as click fraud or referral code abuse. The technical implementation of AI modules can be carried out through separate microservices based on Docker and Kubernetes using TensorFlow, PyTorch libraries, or OpenAI API services.

CONCLUSIONS FROM THIS STUDY AND PROSPECTS FOR FURTHER EXPLORATION IN THIS DIRECTION

The following main results were obtained from this study. An analysis of the current state of the problem of task automation in traffic arbitration teams was conducted, revealing that existing solutions such as ZeyBot, Everbot,

Standuply, and Karma Bot do not provide full integration of automation, gamification, and specialisation for the arbitration model. The choice of technologies was justified, including the Python language, the Aiogram framework with its asynchronous processing and FSM support, the MongoDB database with its flexible document-oriented structure, and the Render and Railway cloud platforms for deployment. A functional-structural model of the system was developed, including a functional-block diagram, component diagram, UML class diagram, user interaction flowchart, and activity diagram. A MongoDB database consisting of six collections was designed, ensuring efficient storage and processing of information about users, tasks, task completions, coin transactions, sessions, and referral relationships. The chatbot code was implemented according to the developed architecture, ensuring modularity, asynchrony, and security. Testing confirmed the stable operation of the system under loads of up to one hundred simultaneous users, with an average response time of less than half a second. Information security measures were proposed, including the use of HTTPS, Telegram ID-based authentication, database access restrictions, backup, logging, and protection against injections. The prospects for integrating artificial intelligence elements were outlined, particularly for intelligent traffic management, natural language processing, recommendation systems, and anomaly detection.

The obtained results have practical significance and can be used in real business processes of traffic arbitration teams, as well as in the educational process to demonstrate modern practices of developing intelligent information systems. Further research will be aimed at integrating AI modules, expanding system functionality, and adapting it to other messengers, such as Slack and Discord.

References

1. HubSpot. Available at: https://blog.hubspot.com/marketing
2. What is a Chatbot? The Ultimate Guide. Available at: https://www.drift.com/blog/chatbots-report
3. The Gamification at Work Survey. Available at: https://www.talentlms.com/blog/gamification-survey-results/
4. Zeydoo Team. Available at: https://zeydoo.com/blog/
5. Everbot for Slack. Available at: https://everbot.cz/ai-chat
6. Standuply. Available at: https://standuply.com/
7. Karma Bot. Available at: https://karmabot.chat/
8. Aiogram Framework. Available at: https://docs.aiogram.dev/en/v3.20.0.post0/
9. MongoDB. Available at: https://www.mongodb.com/why-use-mongodb
10. Render Cloud Platform. Available at: https://render.com/
11. Binom API. Available at: https://binom.org/