Innopolis University
st. Universitetskaia, 1
Innopolis,
Respublika Tatarstan (Tatarstan), 420500

Lost&Found

5 september 2025 г.

OVERVIEW

Lost&Found is a Telegram-based travel discovery bot that delivers personalized travel recommendations. It aggregates and enriches data from multiple sources—including Google Places, Wikipedia, and OpenStreetMap—to provide users with not only destination suggestions but also detailed historical context, cultural insights, and media references. This enhanced experience is powered by advanced Natural Language Processing (NLP) algorithms and semantic search techniques.

Objectives

1. To build an intelligent Telegram bot that offers personalized travel recommendations.
2. To aggregate and process data from multiple trusted sources.
3. To enrich travel information with historical, cultural, and media-related details.
4. To deliver a seamless user experience through efficient data retrieval and conversational interactions.

Scope of the Project

Functional Scope

- Data Aggregation: Collect data from APIs such as Google Places, Wikipedia, and OpenStreetMap.
- NLP Processing: Analyze user reviews and travel-related texts for sentiment, topics, and key phrases using state-of-the-art models (e.g., BERT, Sentence-BERT).
- Search and Recommendation: Implement semantic search (BM25, embedding-based retrieval) to match user queries with enriched travel information.
- Telegram Bot Interface: Develop a conversational bot that responds to commands and queries, providing personalized travel recommendations.

Non-Functional Scope

- Performance: Ensure low latency responses (sub-second) for API queries and bot interactions.
- Scalability: Use database indexing and caching to support a growing number of user queries.
- Maintainability: Write modular, well-documented code with clear API endpoints to allow for future enhancements.
- Security: Secure API keys, implement proper data handling and privacy measures, and ensure the bot is safe from common vulnerabilities.

System Architecture and Components

Overall Architecture

Data Collection Layer

- APIs: Integrate with Google Places, Wikipedia, and OpenStreetMap for primary data.
- Parsers: Develop scripts to retrieve and clean data.

Data Processing & Storage

- Data Cleaning: Normalize and deduplicate data.
- Database: Utilize PostgreSQL for structured data and Elasticsearch for efficient full-text search and semantic queries.

NLP and Enrichment Layer

- Sentiment Analysis & Topic Extraction: Use libraries such as spaCy and Transformers.
- Semantic Search: Implement BM25 along with embedding-based techniques (using BERT/Sentence-BERT) to enhance relevance.

Backend API

- Framework: Use FastAPI or Flask to develop RESTful endpoints for data retrieval and recommendation services.
- Integration: Connect the backend with the database and NLP modules.

Telegram Bot Interface

- Bot Framework: Use python-telegram-bot or Aiogram.
- Command Handling: Design a set of commands (e.g., /start, /find, /random) for user interaction.
- User Query Processing: Forward user requests to the backend API and format responses.

Data Flow Diagram

A simplified data flow:

User Query (Telegram Bot) ➜ Backend API ➜ Database/NLP Processing ➜ Aggregated and Enriched Data ➜ Response to User (via Telegram Bot)

Detailed Requirements

Functional Requirements

User Authentication

- Basic identification via Telegram account; no complex authentication is required.

Search Functionality

- Accept natural language queries.
- Return results ranked by relevance using a combination of keyword matching and semantic search.

Information Enrichment

- Provide historical facts, cultural details, and media references with each recommendation.

Command Set for the Bot:

/start: Introduction and usage guidance.

/find [query]: Search for travel destinations based on the provided query.

/random: Return a randomly selected recommendation.

Optional commands for filters or category-based searches.

Non-Functional Requirements

Performance - API responses should be delivered within 500ms under normal conditions.
Scalability - Support concurrent requests and plan for horizontal scaling if user volume increases.
Reliability - Maintain a high uptime (99.5%+) with monitoring and automated alerts.
Usability - Clear, concise interactions via Telegram; informative error messages and guidance.
Security:

- Secure storage of API keys and user data.
- Implement rate limiting to prevent abuse.

Team Roles and Responsibilities

- Data Engineer:
  - Set up data pipelines.
  - Develop and maintain data parsers.
  - Oversee database design and management.
- NLP Engineer:
  - Implement NLP algorithms for sentiment analysis, topic extraction, and semantic search.
  - Optimize and fine-tune models using real-world travel texts and user reviews.
- Backend Developer:
  - Design and implement RESTful APIs.
  - Ensure secure and efficient communication with the database and NLP modules.
  - Manage server deployment and continuous integration.
- Bot Developer:
  - Develop the Telegram bot interface.
  - Integrate the bot with backend APIs.
  - Design intuitive command structures and conversation flows.

Project Timeline

Total Duration: 12 weeks (Feb 5 – April 30)

- Weeks 1:
  - Requirements gathering, technology stack selection, and finalizing team roles.
- Weeks 2–3:
  - Data collection: Develop parsers, clean data, set up databases.
- Weeks 4–5:
  - NLP module development: Implement and test sentiment analysis and semantic search.
- Weeks 6–8:
  - Backend API development: Create endpoints, integrate with NLP and database.
- Weeks 9–10:
  - Telegram bot development: Build bot interface and integrate API endpoints.
- Weeks 11–12:
  - Testing, debugging, performance optimization, and final refinements.

Risk Management and Quality Assurance

Risk Management

Data Source Reliability:

- Mitigation: Identify backup APIs and establish data caching strategies.

Integration Complexity:

- Mitigation: Conduct regular integration tests and adopt agile methodologies.

Timeline Delays:

- Mitigation: Build in buffer time and monitor progress with daily stand-ups.

Quality Assurance

Code Reviews and Unit Testing: Implement continuous integration (CI) pipelines.

User Acceptance Testing (UAT): Conduct beta testing with a select user group and gather feedback.

Performance Testing: Stress test APIs and ensure response times meet performance targets.

Deliverables and Documentation

Deliverables

Fully functional WanderIntel Telegram bot.

Backend API and data processing modules.

Comprehensive documentation covering system architecture, API usage, and user guides.

Test reports and quality assurance documentation.

Documentation

Technical design document detailing architecture and module specifications.

User manual and quick-start guide for the Telegram bot.

Maintenance and troubleshooting guides.