Σελίδα τίτλου



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Υλοποίηση Έξυπνου Chatbot Εξυπηρέτησης Πελατών με Retrieval-Augmented Generation και Ανατροφοδότηση Ανθρώπου (Human-in-the-Loop)

ΔΗΜΗΤΡΙΟΣ ΤΣΙΛΙΓΙΑΝΝΗΣ

Επιβλέπων: ΦΩΤΙΟΣ ΒΑΡΤΖΙΩΤΗΣ

τίτλος, βαθμίδα

ΑΡΤΑ, ΙΟΥΝΙΟΣ, 2026

Design and Implementation of an Intelligent Customer Service Chatbot with Retrieval-Augmented Generation and Human Feedback (Human-in-the-Loop)

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Όνομα Επίθετο,

2. Μέλος επιτροπής

   Όνομα Επίθετο,

3. Μέλος επιτροπής

   Όνομα Επίθετο,

© ΤΣΙΛΙΓΙΑΝΝΗΣ, ΔΗΜΗΤΡΙΟΣ, 2026.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

ΤΣΙΛΙΓΙΑΝΝΗΣ, ΔΗΜΗΤΡΙΟΣ



ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Βαρτζιώτη Φώτιο, για την καθοδήγηση και τις πολύτιμες παρατηρήσεις του καθ' όλη τη διάρκεια συγγραφής της παρούσας εργασίας. Ευχαριστώ επίσης την οικογένειά μου για την υπομονή και τη συμπαράστασή τους σε όλη αυτή την προσπάθεια.

ΠΕΡΙΛΗΨΗ

Η παρούσα εργασία παρουσιάζει τον σχεδιασμό και την υλοποίηση του ShopEasy ChatBot, ενός έξυπνου συστήματος εξυπηρέτησης πελατών για ηλεκτρονικά καταστήματα, βασισμένου σε τεχνητή νοημοσύνη. Το σύστημα ενσωματώνει δύο προηγμένες τεχνικές: Retrieval-Augmented Generation (RAG) για ανάκτηση σχετικής πληροφορίας από γνωσιακή βάση με χρήση διανυσματικής αναζήτησης, και έναν μηχανισμό Human-in-the-Loop (HitL) ανατροφοδότησης που επιτρέπει τη συνεχή βελτίωση του συστήματος χωρίς επανεκπαίδευση του γλωσσικού μοντέλου. Η αρχιτεκτονική του συστήματος ακολουθεί το τριεπίπεδο μοντέλο: React frontend, Node.js/Express backend και MongoDB Atlas ως βάση δεδομένων με ενσωματωμένη υποστήριξη διανυσματικής αναζήτησης. Για την παραγωγή embeddings χρησιμοποιείται τοπικό μοντέλο (paraphrase-multilingual-MiniLM-L12-v2) χωρίς εξάρτηση από cloud API, ενώ η υποστήριξη πολλαπλών παρόχων AI (OpenAI, Gemini, OpenRouter) επιτυγχάνεται μέσω factory pattern. Η αξιολόγηση πραγματοποιήθηκε με τρία πειράματα σε σύνολο 30 ερωτήσεων τεσσάρων κατηγοριών. Ο μηχανισμός RAG πέτυχε hit rate 93,3%, ενώ η σύγκριση του πλήρους συστήματος με baseline χωρίς RAG έδειξε βελτίωση ακρίβειας κατά 53,3 ποσοστιαίες μονάδες (96,7% έναντι 43,4%). Ο μηχανισμός HitL επαλήθευσε την ικανότητα του συστήματος να ενσωματώνει διορθώσεις σε πραγματικό χρόνο. Η εργασία καταδεικνύει ότι η συνδυαστική χρήση RAG και HitL αποτελεί αποτελεσματική προσέγγιση για την κατασκευή αξιόπιστων chatbots σε επιχειρησιακό πλαίσιο, χωρίς την ανάγκη για εκπαίδευση εξειδικευμένου μοντέλου.

Λέξεις-κλειδιά: Chatbot, Retrieval-Augmented Generation (RAG), Human-in-the-Loop, Embeddings, MongoDB Atlas

ABSTRACT

This thesis presents the design and implementation of ShopEasy ChatBot, an AI-powered customer support system for e-commerce applications. The system integrates two advanced techniques: Retrieval-Augmented Generation (RAG) for retrieving relevant information from a knowledge base using vector similarity search, and a Human-in-the-Loop (HitL) feedback mechanism that enables continuous improvement without retraining the underlying language model. The system follows a three-tier architecture: a React frontend, a Node.js/Express backend, and MongoDB Atlas as the database with native vector search support. Embeddings are generated locally using the paraphrase-multilingual-MiniLM-L12-v2 model, eliminating cloud API dependencies, while multi-provider AI support (OpenAI, Gemini, OpenRouter) is achieved through a factory pattern abstraction. Evaluation was conducted through three experiments on a test set of 30 questions across four categories. The RAG retrieval mechanism achieved a hit rate of 93.3%, while comparison of the full system against a RAG-free baseline demonstrated an accuracy improvement of 53.3 percentage points (96.7% vs. 43.4%). The HitL feedback loop successfully incorporated real-time corrections, validating the inference-time alignment approach. The findings demonstrate that combining RAG with a HitL feedback mechanism is an effective strategy for building reliable, domain-specific chatbots without the need for fine-tuning a dedicated language model.

Keywords: Chatbot, Retrieval-Augmented Generation (RAG), Human-in-the-Loop, Embeddings, MongoDB Atlas

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΥΧΑΡΙΣΤΙΕΣ 6

ΠΕΡΙΛΗΨΗ 7

ABSTRACT 8

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ 9

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ 15

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ 17

ΑΠΟΔΟΣΗ ΟΡΩΝ / ΓΛΩΣΣΑΡΙΟ 18

1. Εισαγωγή 22

1.1 Γενικό Πλαίσιο και Κίνητρο της Έρευνας 22

1.2 Ερευνητικά Ερωτήματα 23

1.3 Σκοπός, Στόχοι και Αναμενόμενη Συνεισφορά 24

1.4 Όρια και Περιορισμοί της Έρευνας 25

1.5 Επιλογή Θέματος: Ακαδημαϊκή και Επαγγελματική Σκοπιμότητα 25

1.6 Δομή της Εργασίας 26

2. Θεωρητικό Υπόβαθρο 27

2.1 Εξέλιξη της Επεξεργασίας Φυσικής Γλώσσας: Από τους Κανόνες στα Νευρωνικά Δίκτυα 27

2.2 Μεγάλα Γλωσσικά Μοντέλα (Large Language Models, LLMs) 28

2.2.1 Η Αρχιτεκτονική Transformer 28

2.2.2 Προ-εκπαίδευση και Κλίμακα 29

2.2.3 Δομικά Προβλήματα των LLMs σε Επιχειρησιακό Πλαίσιο 29

2.3 Αναπαράσταση Κειμένου ως Διανύσματα: Embeddings και Σημασιολογική Ομοιότητα .. 30

2.3.1 Τι είναι τα Embeddings; 30

2.3.2 Ομοιότητα Συνημιτόνου (Cosine Similarity) 30

2.4 Αρχιτεκτονική Retrieval-Augmented Generation (RAG) 31
2.4.1 Ορισμός και Ιστορική Αναδρομή 31
2.4.2 Τα Στάδια του RAG Pipeline 32
2.4.3 RAG έναντι Fine-Tuning: Συγκριτική Ανάλυση 32
2.5 Ευθυγράμμιση Μοντέλων: RLHF και Human-in-the-Loop 33
2.5.1 Το Πρόβλημα της Ευθυγράμμισης (Alignment Problem) 33
2.5.2 Reinforcement Learning from Human Feedback (RLHF) 33
2.5.3 Inference-Time Alignment: Η Προσέγγιση της Παρούσας Εργασίας 34
2.6 Διανυσματικές Βάσεις Δεδομένων και MongoDB Atlas Vector Search 35
2.6.1 Η Ανάγκη για Εξειδικευμένη Αποθήκευση Διανυσμάτων 35
2.6.2 MongoDB Atlas $vectorSearch 35
2.6.3 Πλεονεκτήματα της Ενοποιημένης Προσέγγισης 36
2.7 Συνοψίζοντας: Η Σύνθεση των Τεχνολογιών 36
3. Ανάλυση Αγοράς και Ανταγωνιστικό Τοπίο 36
3.1 Το Μέγεθος και η Δυναμική της Αγοράς 37
3.1.1 Παγκόσμιοι Αριθμοί 37
3.1.2 Αποτυχίες και Ανεκπλήρωτες Προσδοκίες 37
3.2 Κυρίαρχες Εμπορικές Πλατφόρμες: Αναλυτική Κριτική 38
3.3 Συγκριτική Αξιολόγηση 38
3.4 Τάσεις Καταναλωτών και Προσδοκίες για το 2025 39
3.5 Κενά της Αγοράς και Αιτιολόγηση της Παρούσας Υλοποίησης 40
3.6 Σύνοψη Κεφαλαίου 41
4. Μεθοδολογία και Αρχιτεκτονικές Αποφάσεις 41
4.1 Μεθοδολογική Προσέγγιση 41
4.2 Επιλογή Τεχνολογικού Στοιβάγματος 42
4.2.1 Backend: Node.js με Typescript 42
4.2.2 Frontend: React με Typescript και Tailwind CSS 43
4.2.3 Βάση Δεδομένων: MongoDB Atlas 43

4.3 Επιλογή Μοντέλου Embeddings 44
4.3.1 Cloud APIs έναντι Τοπικών Μοντέλων 44
4.3.2 Επιλογή Συγκεκριμένου Μοντέλου 44
4.4 Αφαίρεση Παρόχου AI (Provider Abstraction) 45
4.4.1 Το AIProvider Interface 45
4.4.2 Υποστηριζόμενοι Πάροχοι 45
4.5 Σχεδιασμός του RAG Pipeline 46
4.5.1 Βασικές Παράμετροι Σχεδιασμού 46
4.5.2 Fallback Στρατηγική: Atlas έναντι Local MongoDB 46
4.6 Σχεδιασμός του Human-in-the-Loop Μηχανισμού 47
4.6.1 Φιλοσοφία Σχεδιασμού 47
4.6.2 Ο Κύκλος Ανατροφοδότησης: Τεχνική Υλοποίηση 47
4.6.3 Τεχνικοί Περιορισμοί και πως Αντιμετωπίστηκαν 48
4.7 Υποδομή: Docker και Περιβαλλοντικές Μεταβλητές 48
4.8 Σύνοψη Αρχιτεκτονικών Αποφάσεων 49
5. Σχεδιασμός Συστήματος 50
5.1 Επισκόπηση Τριεπίπεδης Αρχιτεκτονικής 50
5.2 Δομή Κώδικα 51
5.3 Σχήματα Δεδομένων MongoDB 52
5.3.1 Συλλογή conversations 53
5.3.2 Συλλογή feedbacks 53
5.3.3 Συλλογή knowledgebases 54
5.4 API Contracts 55
5.5 Ροές Δεδομένων 57
5.5.1 Ροή Chat Request (End-to-End) 57
5.5.2 Ροή Feedback Loop (End-to-End) 58
5.5.3 Ροή Διαχείρισης Γνωσιακής Βάσης 60
5.6 Σχεδιασμός Dynamic System Prompt 61

5.7 Σύνοψη Κεφαλαίου ... 62
6. Υλοποίηση ... 62
6.1 Embedding Service: Τοπική Παραγωγή Διανυσμάτων ... 62
6.1.1 Αποτύπωμα Μοντέλου και Αποθήκευση Docker Volume ... 63
6.2 RAG Pipeline: Από την Ερώτηση στο Context ... 64
6.2.1 Vector Search (Atlas) ... 64
6.2.2 Παράμετροι Απόδοσης ... 65
6.2.3 Fallback σε Local MongoDB ... 66
6.3 Chat Service: Ενορχήστρωση της Πλήρους Ροής ... 66
6.3.1 Κατασκευή Δυναμικού System Prompt ... 66
6.4 Feedback Engine: Inference-Time Alignment ... 67
6.4.1 GetGoldenRules και GetNegativeExamples ... 67
6.4.2 Μορφοποίηση BuildFeedbackPromptSection ... 68
6.5 Αφαίρεση Παρόχου AI (Factory Pattern) ... 68
6.5.1 Το AIProvider Interface ... 69
6.5.2 Διαχείριση Ιστορικού Συνομιλίας Ανά Πάροχο ... 69
6.6 Chat Interface: Frontend Υλοποίηση ... 70
6.6.1 Session Management και Persistence ... 72
6.6.2 Two-Step Clear και Error Banner ... 72
6.6.3 FeedbackButtons: Κλιμακωτή Υποβολή ... 72
6.7 Admin Panel: Πλήρης Κύκλος Διαχείρισης ... 73
6.7.1 Dashboard Tab: Workflow Εγκρίσεων ... 74
6.7.2 Knowledge Base Tab: Live Embedding Updates ... 75
6.8 Seeding Γνωσιακής Βάσης ... 76
6.9 Infrastructure: Docker Compose ... 76
6.10 Τεχνικές Προκλήσεις και Εμπειρίες Υλοποίησης ... 77
6.11 Σύνοψη Κεφαλαίου ... 78
7. Αξιολόγηση ... 79

7.1 Μεθοδολογία Αξιολόγησης 79
7.2 Σχεδιασμός Test Set 80
7.2.1 Οι Τέσσερις Κατηγορίες Ερωτήσεων 80
7.2.2 Παρουσίαση του Test Set 81
7.3 Πείραμα 1: Αξιολόγηση του RAG Retrieval 81
7.3.1 Αποτελέσματα Hit Rate 81
7.3.2 Ανάλυση των Δύο Αποτυχιών Ανάκτησης 82
7.3.3 Το Φαινόμενο του 0% Off-topic Rejection 83
7.4 Πείραμα 2: Σύγκριση Full System έναντι Baseline 84
7.4.1 Κριτήρια Βαθμολόγησης 85
7.4.2 Συνολικά Αποτελέσματα 85
7.4.3 Ανάλυση ανά Κατηγορία 85
7.4.4 Τυπικές Ψευδαισθήσεις του Baseline 86
7.4.5 Οι Δύο Αποτυχίες του Full-chat 87
7.5 Πείραμα 3: Επικύρωση του Feedback Loop 87
7.5.1 Μεθοδολογία του Πειράματος 87
7.5.2 Τα Πέντε Σενάρια Δοκιμής 87
7.5.3 Αποτελέσματα 88
7.5.4 Το Σενάριο 4: Ο Feedback Loop ως Δίχτυ Ασφαλείας 88
7.5.5 Τεχνική Λειτουργία της Έγχυσης 89
7.6 Συζήτηση Ευρημάτων και Περιορισμοί 89
7.6.1 Περιορισμοί της Αξιολόγησης 90
7.6.2 Αρχιτεκτονικά Συμπεράσματα 90
7.7 Σύνδεση με τα Ερευνητικά Ερωτήματα 91
7.8 Σύνοψη Κεφαλαίου 92
8. Συμπεράσματα και Μελλοντική Δουλειά 92
8.1 Σύνοψη της Εργασίας 92
8.2 Τι Λειτούργησε και τι Όχι 93

8.2.1 Θετικά Αποτελέσματα 93
8.2.2 Περιορισμοί και Αδυναμίες 93
8.3 Συνεισφορά της Εργασίας 94
8.4 Μελλοντική Δουλειά 94
8.5 Κλείσιμο 94
9. Βιβλιογραφία 95
10. Παράρτημα Α: Πλήρες Σύνολο Ελέγχου (30 Ερωτήσεις) 97

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 3.1: Συγκριτική αξιολόγηση εμπορικών πλατφορμών AI εξυπηρέτησης πελατών……39

Πίνακας 3.2: Τοποθέτηση της παρούσας υλοποίησης έναντι εμπορικών λύσεων……40

Πίνακας 4.1: Σύγκριση υποστηριζόμενων παρόχων AI……46

Πίνακας 4.2: Κεντρικές περιβαλλοντικές μεταβλητές και ο ρόλος τους……49

Πίνακας 4.3: Σύνοψη αρχιτεκτονικών αποφάσεων……50

Πίνακας 5.1: Τα τρία επίπεδα του συστήματος……51

Πίνακας 5.2: Δομή αρχείων backend……52

Πίνακας 5.3: Σχήμα συλλογής conversations……53

Πίνακας 5.4: Σχήμα συλλογής feedbacks……54

Πίνακας 5.5: Σχήμα συλλογής knowledgebases……55

Πίνακας 5.6: REST API endpoints……56

Πίνακας 5.7: Βήματα επεξεργασίας chat request……57

Πίνακας 5.8: Στάδια ροής Human-in-the-Loop……59

Πίνακας 5.9: Λογική αυτόματης ενημέρωσης embedding……60

Πίνακας 5.10: Δομή Dynamic System Prompt……61

Πίνακας 6.1: Παράμετροι μοντέλου embeddings……64

Πίνακας 6.2: Παράμετροι RAG pipeline……65

Πίνακας 6.3: Μορφή golden rule στο system prompt……68

Πίνακας 6.4: Σύγκριση format μηνυμάτων ανά πάροχο……69

Πίνακας 6.5: Κατανομή 25 FAQ εγγραφών ανά κατηγορία……76

Πίνακας 6.6: Docker Compose υπηρεσίες……77

Πίνακας 7.1: Αντιπροσωπευτικό δείγμα του test set……81

Πίνακας 7.2: Αποτελέσματα ανάκτησης ανά κατηγορία……81

Πίνακας 7.3: Συμπεριφορά του RAG σε άσχετες ερωτήσεις……83

Πίνακας 7.4: Συνολική ακρίβεια full-chat έναντι baseline……85

Πίνακας 7.5: Ακρίβεια ανά κατηγορία ερώτησης……85

Πίνακας 7.6: Σενάρια του πειράματος feedback loop……87

Πίνακας 7.7: Σύγκριση απαντήσεων πριν και μετά την έγχυση διορθώσεων……88

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ

Διάγραμμα 5.1: Τριεπίπεδη Αρχιτεκτονική του συστήματος ShopEasy ChatBot……51

Διάγραμμα 5.2: Σχήματα δεδομένων MongoDB (Collections)……55

Διάγραμμα 5.3: Ακολουθία επεξεργασίας αιτήματος chat……58

Διάγραμμα 5.4: Μηχανή καταστάσεων feedback document……59

Διάγραμμα 5.5: Κύκλος Human-in-the-Loop ανατροφοδότησης……60

Διάγραμμα 7.1: Σύγκριση Baseline έναντι Full-chat RAG Pipeline……84

Εικόνα 6.1: Αρχική οθόνη του chat interface με μήνυμα καλωσορίσματος……70

Εικόνα 6.2: Παραδείγματα ερωτήσεων χρήστη στο chat interface……71

Εικόνα 6.3: Πίνακας διαχείρισης — Admin Dashboard……73

Εικόνα 6.4: Καρτέλα Γνωσιακής Βάσης στον Admin Panel……75

ΑΠΟΔΟΣΗ ΟΡΩΝ / ΓΛΩΣΣΑΡΙΟ

Το παρόν γλωσσάρι συγκεντρώνει τους κύριους τεχνικούς όρους που χρησιμοποιούνται στην εργασία, με σύντομη επεξήγηση για αναγνώστες που δεν έχουν εξειδικευμένο τεχνικό υπόβαθρο. Οι όροι παρατίθενται αλφαβητικά (αγγλικά).

ANN (Approximate Nearest Neighbor). Οικογένεια αλγορίθμων που βρίσκουν τους πιο κοντινούς γείτονες ενός σημείου σε διανυσματικό χώρο με προσεγγιστικό τρόπο, ανταλλάσσοντας λίγη ακρίβεια για πολύ μεγαλύτερη ταχύτητα έναντι της εξαντλητικής (brute-force) σύγκρισης. Απαραίτητοι για διανυσματική αναζήτηση σε μεγάλες βάσεις.

BM25. Κλασική συνάρτηση κατάταξης για αναζήτηση κειμένου, βασισμένη σε στατιστική λέξεων (term frequency, inverse document frequency, μήκος εγγράφου). Επιστρέφει αποτελέσματα βάσει επικάλυψης λέξεων-κλειδιών, χωρίς κατανόηση νοήματος. Λειτουργεί ως «sparse retrieval» σε αντίθεση με το «dense retrieval» των embeddings.

Cosine Similarity (Ομοιότητα Συνημιτόνου). Μέτρο ομοιότητας δύο διανυσμάτων που υπολογίζει το συνημίτονο της γωνίας μεταξύ τους. Τιμή 1 σημαίνει ταυτόσημη κατεύθυνση (πανομοιότυπο νόημα), 0 σημαίνει ανεξάρτητα διανύσματα, -1 αντίθετο νόημα. Στην παρούσα εργασία χρησιμοποιείται κατώφλι 0,5.

Embedding (Διανυσματική Ενσωμάτωση). Αριθμητική αναπαράσταση κειμένου ως διάνυσμα σε πολυδιάστατο χώρο. Στη συγκεκριμένη υλοποίηση κάθε κείμενο μετατρέπεται σε διάνυσμα 384 διαστάσεων. Σημασιολογικά παρόμοια κείμενα παράγουν γεωμετρικά κοντινά διανύσματα.

Few-shot Prompting. Τεχνική κατά την οποία στο system prompt ενός LLM παρατίθενται λίγα παραδείγματα ζευγών εισόδου-εξόδου, με σκοπό να καθοδηγηθεί η συμπεριφορά του μοντέλου χωρίς επανεκπαίδευση. Στο παρόν σύστημα οι εγκεκριμένες διορθώσεις του Human-in-the-Loop λειτουργούν ως few-shot παραδείγματα.

Fine-tuning (Μικρορύθμιση). Διαδικασία συνέχισης της εκπαίδευσης ενός προεκπαιδευμένου μοντέλου σε εξειδικευμένο σύνολο δεδομένων, ώστε να προσαρμοστεί σε συγκεκριμένη εργασία ή πεδίο. Απαιτεί υπολογιστικούς πόρους και ποιοτικό dataset. Δεν εφαρμόζεται στην παρούσα εργασία.

Hallucination (Ψευδαίσθηση). Φαινόμενο κατά το οποίο ένα LLM παράγει πληροφορίες που ακούγονται πειστικές αλλά είναι εσφαλμένες ή κατασκευασμένες. Σε επιχειρησιακό πλαίσιο αποτελεί κρίσιμο ρίσκο. Η αρχιτεκτονική RAG περιορίζει σημαντικά αυτό το φαινόμενο.

HNSW (Hierarchical Navigable Small World). Αλγόριθμος ANN που βασίζεται σε ιεραρχική δομή γράφων «μικρού κόσμου». Πετυχαίνει λογαριθμική πολυπλοκότητα αναζήτησης, διατηρώντας υψηλό recall. Χρησιμοποιείται εσωτερικά από το MongoDB Atlas Vector Search (Malkov & Yashunin, 2020).

Human-in-the-Loop (HitL). Σχήμα εργασίας όπου ένας άνθρωπος εμπλέκεται σε κρίσιμα σημεία απόφασης ενός αυτοματοποιημένου συστήματος. Στην παρούσα εργασία, διαχειριστής εγκρίνει ή απορρίπτει τις διορθώσεις χρηστών πριν επηρεάσουν τη συμπεριφορά του μοντέλου.

Inference-time Alignment. Τροποποίηση της συμπεριφοράς ενός LLM κατά τον χρόνο εκτέλεσης, χωρίς αλλαγή των παραμέτρων του (συνήθως μέσω prompt engineering ή few-shot examples στο context).

LLM (Large Language Model). Μεγάλο Γλωσσικό Μοντέλο. Νευρωνικό δίκτυο βασισμένο σε αρχιτεκτονική Transformer, με δισεκατομμύρια παραμέτρους, εκπαιδευμένο σε τεράστιο όγκο κειμένου. Παραδείγματα: GPT-4, Gemini, Llama.

NLP (Natural Language Processing). Επεξεργασία Φυσικής Γλώσσας. Διεπιστημονικό πεδίο που εστιάζει στην αυτόματη επεξεργασία και κατανόηση ανθρώπινης γλώσσας από υπολογιστές.

PPO (Proximal Policy Optimization). Αλγόριθμος ενισχυτικής μάθησης που χρησιμοποιείται στη φάση RL Optimization του RLHF (Schulman et al., 2017). Επιτρέπει σταθερή βελτιστοποίηση της πολιτικής του μοντέλου διατηρώντας περιορισμένη απόκλιση από την προηγούμενη.

RAG (Retrieval-Augmented Generation). Ανάκτηση Ενισχυμένη Παραγωγή. Αρχιτεκτονική που συνδυάζει ένα σύστημα ανάκτησης πληροφορίας με ένα γλωσσικό μοντέλο: πριν το μοντέλο παράγει απάντηση, του παρέχονται σχετικά έγγραφα ως context (Lewis et al., 2020· Gao et al., 2024).

Recall και Precision. Δύο βασικές μετρικές αξιολόγησης συστημάτων ανάκτησης. Recall: ποσοστό σχετικών εγγράφων που εντοπίστηκαν. Precision: ποσοστό των εντοπισμένων εγγράφων που είναι όντως σχετικά. Συνήθως υπάρχει trade-off μεταξύ τους.

RLHF (Reinforcement Learning from Human Feedback). Ενισχυτική Μάθηση από Ανθρώπινη Ανατροφοδότηση. Διαδικασία ευθυγράμμισης (alignment) ενός LLM σε τρεις φάσεις: SFT, Reward Modeling, RL Optimization. Χρησιμοποιήθηκε εκτεταμένα στην εκπαίδευση του ChatGPT (Ouyang et al., 2022).

SFT (Supervised Fine-Tuning). Πρώτη φάση του RLHF, όπου το μοντέλο εκπαιδεύεται μέσω επιβλεπόμενης μάθησης σε υποδειγματικές απαντήσεις γραμμένες από ανθρώπους.

Semantic Search (Σημασιολογική Αναζήτηση). Αναζήτηση που βασίζεται στη νοηματική ομοιότητα κειμένων και όχι σε επικάλυψη λέξεων. Υλοποιείται με embeddings και μετρικές όπως η ομοιότητα συνημιτόνου.

Sentence-BERT (SBERT). Παραλλαγή του BERT με σιαμαία αρχιτεκτονική, σχεδιασμένη για παραγωγή σημασιολογικά ουσιαστικών embeddings ολόκληρων προτάσεων. Το paraphrase-multilingual-MiniLM-L12-v2 που χρησιμοποιείται στην παρούσα υλοποίηση είναι παράγωγο της οικογένειας SBERT (Reimers & Gurevych, 2019).

Soft Delete. Λογική διαγραφή, όπου μια εγγραφή σημαίνεται ως ανενεργή (π.χ. isActive=false) αντί να αφαιρείται φυσικά από τη βάση δεδομένων. Επιτρέπει ανάκτηση και διατήρηση ιστορικού.

System Prompt. Σύστημα οδηγιών που δίνεται σε ένα LLM πριν από την αλληλεπίδραση με τον χρήστη και καθορίζει τον ρόλο, τη συμπεριφορά και τους περιορισμούς του. Στο σύστημα της εργασίας το system prompt είναι δυναμικό και ανακατασκευάζεται σε κάθε αίτημα.

Token. Βασική μονάδα κειμένου που επεξεργάζεται ένα LLM, συχνά αντιστοιχεί σε λέξη, υπολέξη ή σύμβολο. Το κόστος των APIs LLM χρεώνεται συνήθως ανά token.

Transformer. Αρχιτεκτονική νευρωνικού δικτύου βασισμένη σε μηχανισμό self-attention, που αποτελεί τη βάση όλων των σύγχρονων LLMs (Vaswani et al., 2017).

Vector Database (Διανυσματική Βάση Δεδομένων). Σύστημα διαχείρισης δεδομένων εξειδικευμένο στην αποθήκευση και αναζήτηση διανυσμάτων υψηλής διαστατικότητας. Παραδείγματα: MongoDB Atlas Vector Search (στην παρούσα εργασία), Pinecone, Weaviate, Chroma.

Word2Vec. Πρώιμη και ευρέως γνωστή μέθοδος παραγωγής word embeddings (Mikolov et al., 2013). Αν και ξεπεράστηκε από νεότερα μοντέλα όπως το BERT και το Sentence-BERT, παραμένει βασική αναφορά για την κατανόηση των διανυσματικών αναπαραστάσεων λέξεων.

1. Εισαγωγή

1.1 Γενικό Πλαίσιο και Κίνητρο της Έρευνας

Το ηλεκτρονικό εμπόριο αναπτύσσεται με γρήγορους ρυθμούς. Σύμφωνα με τα στοιχεία της Statista (2024), ο παγκόσμιος τζίρος του ηλεκτρονικού λιανεμπορίου ξεπέρασε τα 5,8 τρισεκατομμύρια δολάρια το 2023, με αναμενόμενη αύξηση άνω του 39% έως το 2027. Η ανάπτυξη αυτή δεν συνοδεύτηκε από ανάλογη αύξηση στελέχωσης των τμημάτων εξυπηρέτησης πελατών, με αποτέλεσμα μεγάλους χρόνους αναμονής και μεταβλητή ποιότητα εξυπηρέτησης.

Η αυτοματοποίηση της εξυπηρέτησης πελατών μέσω λογισμικού δεν αποτελεί νέα ιδέα. Από τα απλά συστήματα Interactive Voice Response (IVR) της δεκαετίας του 1970 έως τα rule-based chatbots του 2010, η βιομηχανία επιδίωκε πάντοτε να μειώσει το λειτουργικό κόστος διατηρώντας ένα αποδεκτό επίπεδο ποιότητας εξυπηρέτησης. Ωστόσο, οι παραδοσιακές αυτές προσεγγίσεις χαρακτηρίζονταν από κοινά δομικά μειονεκτήματα: αδυναμία κατανόησης φυσικής γλώσσας, ανεπαρκής διαχείριση πολυσύνθετων ή διφορούμενων ερωτημάτων, και αδυναμία προσαρμογής σε νέες πληροφορίες χωρίς πλήρη επαναπρογραμματισμό.

Η εμφάνιση των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models, LLMs), και ιδιαίτερα η δημόσια κυκλοφορία του ChatGPT τον Νοέμβριο του 2022, άλλαξε αρκετά τις δυνατότητες αυτοματοποίησης. Τα LLMs επιδεικνύουν ικανότητες κατανόησης και παραγωγής φυσικής γλώσσας, επιτρέποντας τη διεξαγωγή συνομιλιών ανεξάρτητα από προκαθορισμένα μονοπάτια διαλόγου. Παρά το τεράστιο αυτό δυναμικό, η άμεση ενσωμάτωσή τους σε επιχειρησιακό περιβάλλον αποκαλύπτει δομικά προβλήματα που καθιστούν την απλή χρήση τους ανεπαρκή: τείνουν να παράγουν πληροφορίες με βεβαιότητα ακόμα και όταν αυτές είναι εσφαλμένες (φαινόμενο που ονομάζεται «ψευδαίσθηση» ή hallucination), δεν έχουν πρόσβαση σε ιδιωτικά δεδομένα επιχειρήσεων που δεν συμπεριλήφθηκαν στην εκπαίδευσή τους, και η συμπεριφορά τους είναι εγγενώς μη-ντετερμινιστική, καθιστώντας δύσκολο τον έλεγχο της ποιότητας των απαντήσεων.

Την ίδια περίοδο, η αγορά των έτοιμων λύσεων εξυπηρέτησης πελατών με τεχνητή νοημοσύνη (AI-powered SaaS) έχει αναπτυχθεί με ταχείς ρυθμούς, προσφέροντας πλατφόρμες όπως οι Intercom Fin, Zendesk AI και Salesforce Einstein. Αλλά αυτές οι λύσεις παρουσιάζουν σημαντικά μειονεκτήματα για επιχειρήσεις μικρού και μεσαίου μεγέθους: λειτουργούν ως «μαύρα κουτιά» (black boxes) με ελάχιστη διαφάνεια στον εσωτερικό μηχανισμό τους, επιβάλλουν απρόβλεπτα μοντέλα κοστολόγησης (π.χ. χρέωση ανά επιτυχή επίλυση), και εγείρουν σοβαρά ζητήματα κυριαρχίας δεδομένων (κυριαρχία δεδομένων), ιδιαίτερα σε πλαίσιο εφαρμογής του Γενικού Κανονισμού Προστασίας Δεδομένων (GDPR) της Ευρωπαϊκής Ένωσης.

Από τη μία πλευρά υπάρχει η ανάγκη για έξυπνη αυτοματοποίηση της εξυπηρέτησης πελατών· από την άλλη, για τις μικρές και μεσαίες επιχειρήσεις λείπει μια ανοιχτή, ελεγχόμενη και οικονομικά βιώσιμη επιλογή. Σε αυτό το κενό απαντά το πρωτότυπο της παρούσας εργασίας, αξιοποιώντας τις τεχνικές RAG και Human-in-the-Loop feedback.

1.2 Ερευνητικά Ερωτήματα

Η έρευνα δομείται γύρω από τρία κεντρικά ερευνητικά ερωτήματα, τα οποία ορίζουν το πεδίο διερεύνησης και αποτελούν το κριτήριο επιτυχίας της υλοποίησης. Τα ερωτήματα αυτά επιλέχθηκαν έτσι ώστε να καλύπτουν συνολικά τόσο την τεχνική εφικτότητα όσο και την επιχειρησιακή βιωσιμότητα της προτεινόμενης αρχιτεκτονικής.

RQ1: Σε ποιο βαθμό μπορεί μια ελαφριά αρχιτεκτονική Ανάκτησης Ενισχυμένης Παραγωγής (RAG), που βασίζεται σε τοπικά παραγόμενα διανύσματα (embeddings) και σε σημασιολογική αναζήτηση μέσω MongoDB Atlas, να περιορίσει αποτελεσματικά τις ψευδαισθήσεις ενός Μεγάλου Γλωσσικού Μοντέλου και να παρέχει έγκυρες, εστιασμένες απαντήσεις σε εξειδικευμένα επιχειρησιακά ερωτήματα, χωρίς την ανάγκη υπολογιστικά ακριβούς μικρορύθμισης (fine-tuning);

RQ2: Πώς μπορεί ένας μηχανισμός ανθρώπινης ανατροφοδότησης (Human-in-the-Loop) να αξιοποιηθεί κατά τον χρόνο εκτέλεσης (inference time), μέσω δυναμικής έγχυσης εγκεκριμένων διορθώσεων στο σύστημα οδηγιών (system prompt) ως παραδείγματα λίγων δειγμάτων (few-shot examples), για να επιτύχει άμεση συμπεριφορική διόρθωση του μοντέλου, προσομοιώνοντας τη φάση Μοντελοποίησης Ανταμοιβής (Reward Modeling) της RLHF χωρίς επανεκπαίδευση;

RQ3: Είναι τεχνικά και οικονομικά βιώσιμη μια custom open-source αρχιτεκτονική, βασισμένη σε ενοποιημένο επίπεδο δεδομένων (MongoDB Atlas) για ταυτόχρονη αποθήκευση συνομιλιών, διανυσμάτων και ανατροφοδοτήσεων, έναντι των καθιερωμένων εμπορικών λύσεων AI εξυπηρέτησης πελατών;

Τα τρία ερωτήματα συνδέονται μεταξύ τους. Δεν έχει νόημα να μιλήσουμε για αποτελεσματικό feedback loop αν το RAG δεν επιστρέφει σχετικά αποτελέσματα από την αρχή· και η συζήτηση για βιωσιμότητα έναντι των εμπορικών λύσεων στηρίζεται στο αν τα δύο πρώτα όντως δουλεύουν.

1.3 Σκοπός, Στόχοι και Αναμενόμενη Συνεισφορά

Ο κύριος σκοπός της εργασίας είναι ο σχεδιασμός, η υλοποίηση και η αξιολόγηση του «ShopEasy ChatBot», ενός λειτουργικού πρωτοτύπου συστήματος εξυπηρέτησης πελατών για ένα εικονικό ηλεκτρονικό κατάστημα. Το πρωτότυπο αποτελεί εφαρμοσμένη ερευνητική πλατφόρμα μέσα από την οποία εξετάζονται, σε πραγματικές συνθήκες εκτέλεσης, οι δυνατότητες και τα όρια των τεχνολογιών RAG και Human-in-the-Loop.

Για την επίτευξη του παραπάνω σκοπού, τέθηκαν οι ακόλουθοι επιμέρους στόχοι:

• **Θεωρητική τεκμηρίωση**: Συστηματική ανασκόπηση της βιβλιογραφίας αναφορικά με τα LLMs, την αρχιτεκτονική RAG, τις μεθόδους ευθυγράμμισης μοντέλων (RLHF, RLAIF) και τις συγκριτικές τάσεις της αγοράς εξυπηρέτησης πελατών με AI.

• **Αρχιτεκτονικός σχεδιασμός**: Ανάπτυξη τριεπίπεδης αρχιτεκτονικής (3-tier) με σαφή διαχωρισμό επιπέδου παρουσίασης (React), επιχειρησιακής λογικής (Node.js/Express) και δεδομένων (MongoDB Atlas), με έμφαση στην αντικαταστασιμότητα του παρόχου AI μέσω μοτίβου εργοστασίου (Factory Pattern).

• **Υλοποίηση RAG pipeline**: Ενσωμάτωση τοπικής παραγωγής διανυσμάτων (χωρίς cloud API) με χρήση πολύγλωσσου μοντέλου, και σημασιολογική αναζήτηση μέσω MongoDB Atlas $vectorSearch.

• **Υλοποίηση Human-in-the-Loop**: Ανάπτυξη πλήρους κύκλου ανατροφοδότησης, από τη συλλογή αρνητικών αξιολογήσεων χρηστών έως την έγκριση διορθώσεων από διαχειριστή και την αυτόματη έγχυσή τους στο σύστημα οδηγιών.

• **Αξιολόγηση**: Σχεδιασμός και εκτέλεση σεναρίων δοκιμών (test scenarios) που αποδεικνύουν την αποτελεσματικότητα του συστήματος σε σχέση με τα ερευνητικά ερωτήματα.

Η εργασία συνεισφέρει σε τρεις κατευθύνσεις. Συγκρίνει πρακτικά RAG και fine-tuning σε περιβάλλον με περιορισμένους πόρους. Δείχνει ότι η ευθυγράμμιση κατά την εκτέλεση (inference-

time alignment) μπορεί να λειτουργήσει ικανοποιητικά χωρίς επανεκπαίδευση. Τέλος, ο κώδικας του συστήματος είναι ανοιχτός και μπορεί να αποτελέσει βάση για μελλοντικές επεκτάσεις.

1.4 Όρια και Περιορισμοί της Έρευνας

Είναι χρήσιμο να οριστούν από νωρίς τα όρια της εργασίας. Δεν πρόκειται για αδυναμίες αλλά για συνειδητές επιλογές που διατηρούν τη θεματική εστίαση.

Ως προς το θεματικό πεδίο, το σύστημα περιορίζεται στη διαχείριση κειμενικών αλληλεπιδράσεων (text-based) σε πέντε προκαθορισμένους θεματικούς άξονες: πολιτική επιστροφών προϊόντων, αποστολές και παραδόσεις, μέθοδοι πληρωμής, πληροφορίες προϊόντων και διαχείριση λογαριασμού. Η διαχείριση πολυτροπικών δεδομένων (Multimodal), π.χ. ανάλυση εικόνων βλάβης προϊόντων, εξετάζεται ως πρόταση μελλοντικής επέκτασης αλλά δεν υλοποιείται στο παρόν πρωτότυπο.

Ως προς τον ρόλο της τεχνητής νοημοσύνης, το σύστημα δεν εκτελεί ενεργές ενέργειες σε τρίτα συστήματα (Function Calling / Tool Use). Δεν συνδέεται, δηλαδή, με βάσεις δεδομένων παραγγελιών για να ελέγξει την κατάσταση μιας αποστολής ή να εκκινήσει διαδικασία επιστροφής, απαντά αποκλειστικά σε ερωτήσεις πολιτικής βάσει της γνωσιακής βάσης. Αυτός ο περιορισμός ήταν συνειδητός, ώστε το σύστημα να παραμένει επαληθεύσιμο.

Ως προς την αξιολόγηση, η εργασία δεν διεξάγει συγκριτικές δοκιμές με εμπορικές πλατφόρμες (π.χ. Intercom Fin) σε κοινό περιβάλλον, καθώς η πρόσβαση σε αυτές απαιτεί εμπορικές συμφωνίες. Η σύγκριση με τις εμπορικές λύσεις γίνεται βάσει δημοσιευμένων τεχνικών εγγράφων, ανεξάρτητων ερευνών και τεκμηριωμένων στοιχείων κόστους.

Ως προς τη γλώσσα, η γνωσιακή βάση του συστήματος είναι γραμμένη στα ελληνικά, ενώ το μοντέλο παραγωγής διανυσμάτων υποστηρίζει διαγλωσσική αναζήτηση (cross-lingual retrieval). Η αξιολόγηση, ωστόσο, επικεντρώνεται σε ελληνικές ερωτήσεις χρηστών, και δεν διεξάγεται συστηματική αξιολόγηση πολυγλωσσικής απόδοσης.

Τέλος, το σύστημα σχεδιάστηκε για τοπική ανάπτυξη (local deployment) μέσω Docker και δεν περιλαμβάνει παραγωγική υποδομή (production infrastructure) με αρχιτεκτονική υψηλής διαθεσιμότητας, load balancing ή CI/CD pipeline. Η επιλογή αυτή είναι σκόπιμη: ο στόχος είναι η επίδειξη αρχιτεκτονικής ορθότητας και ερευνητικής επαλήθευσης, όχι η παραγωγή προϊόντος.

1.5 Επιλογή Θέματος: Ακαδημαϊκή και Επαγγελματική Σκοπιμότητα

Το θέμα επιλέχθηκε με βάση δύο παρατηρήσεις. Από ακαδημαϊκής σκοπιάς, η ταχεία εξέλιξη των LLMs δημιουργεί ένα ζωντανό ερευνητικό πεδίο όπου τα βασικά ερωτήματα, πώς μαθαίνουν τα μοντέλα από ανθρώπινη ανατροφοδότηση, πώς αποφεύγονται οι ψευδαισθήσεις, πώς

ενσωματώνεται γνώση σε πραγματικό χρόνο, παραμένουν ανοιχτά και ερευνητικά ενεργά. Η παρούσα εργασία τοποθετείται στην τομή αυτών των ερωτημάτων, εφαρμόζοντάς τα σε ένα ρεαλιστικό επιχειρησιακό σενάριο.

Από επαγγελματικής σκοπιάς, η εξυπηρέτηση πελατών αποτελεί τον τομέα με τον υψηλότερο ρυθμό υιοθέτησης λύσεων τεχνητής νοημοσύνης διεθνώς. Σύμφωνα με πρόβλεψη της Gartner (2022), έως το 2027 τα chatbots αναμένεται να αποτελέσουν το κύριο κανάλι εξυπηρέτησης για περίπου το ένα τέταρτο των οργανισμών. Η κατανόηση της αρχιτεκτονικής, των δυνατοτήτων και των ορίων αυτών των συστημάτων δεν αποτελεί απλώς ακαδημαϊκή άσκηση, αποτελεί επαγγελματικό εφόδιο με πρακτική αξία στην τρέχουσα αγορά εργασίας.

Η επιλογή του εικονικού καταστήματος «ShopEasy» ως περιβάλλον αξιολόγησης έγινε με συγκεκριμένα κριτήρια. Το e-commerce αποτελεί τομέα με καλά τεκμηριωμένες και μετρήσιμες κατηγορίες ερωτημάτων (επιστροφές, αποστολές, πληρωμές), γεγονός που επιτρέπει τη συστηματική αξιολόγηση της απόδοσης του συστήματος. Ταυτόχρονα, η πολυπλοκότητά του είναι επαρκής για να αναδείξει τα πλεονεκτήματα της RAG αρχιτεκτονικής έναντι απλής keyword-based αναζήτησης, χωρίς να καθιστά απαγορευτικά πολύπλοκη την υλοποίηση.

1.6 Δομή της Εργασίας

Η παρούσα εργασία οργανώνεται σε εννέα κεφάλαια, καθένα από τα οποία επιτελεί συγκεκριμένο ρόλο στην ολοκλήρωση της ερευνητικής αφήγησης.

Το παρόν Κεφάλαιο 1 (**Εισαγωγή**) ορίζει το πλαίσιο, το κίνητρο, τα ερευνητικά ερωτήματα, τους στόχους και τα όρια της εργασίας. Αποτελεί τον οδικό χάρτη που καθοδηγεί τον αναγνώστη στα επόμενα κεφάλαια.

Το Κεφάλαιο 2 (**Θεωρητικό Υπόβαθρο**) παρέχει τη βιβλιογραφική τεκμηρίωση των τεχνολογιών που χρησιμοποιούνται: αρχιτεκτονική Transformer, LLMs, διανύσματα και σημασιολογική ομοιότητα, αρχιτεκτονική RAG, τεχνικές ευθυγράμμισης (RLHF, RLAIF) και διανυσματικές βάσεις δεδομένων.

Το Κεφάλαιο 3 (**Ανάλυση Αγοράς**) εξετάζει συγκριτικά τις κυρίαρχες εμπορικές πλατφόρμες AI εξυπηρέτησης πελατών, αναδεικνύει τα κενά τους και τεκμηριώνει την ανάγκη για custom λύσεις.

Το Κεφάλαιο 4 (**Μεθοδολογία και Αρχιτεκτονικές Αποφάσεις**) παρουσιάζει και αιτιολογεί τις κεντρικές σχεδιαστικές επιλογές: RAG έναντι fine-tuning, MongoDB έναντι εξειδικευμένων vector databases, τοπικά έναντι cloud embeddings, και αφαίρεση παρόχου AI (αφαίρεση παρόχου).

Το Κεφάλαιο 5 (**Σχεδιασμός Συστήματος**) αναλύει την αρχιτεκτονική σε επίπεδο λεπτομέρειας: σχήματα δεδομένων, API contracts, διαγράμματα ροής δεδομένων και δομή του κώδικα.

Το Κεφάλαιο 6 (**Υλοποίηση**) περιγράφει τον τρόπο υλοποίησης των κεντρικών components: RAG pipeline, feedback loop, admin panel και chat interface, με παρουσίαση επιλεγμένων τμημάτων κώδικα και σχολιασμό των αποφάσεων υλοποίησης.

Το Κεφάλαιο 7 (**Αξιολόγηση**) παρουσιάζει τα αποτελέσματα σεναρίων δοκιμών, ποιοτικής σύγκρισης (με και χωρίς RAG/Feedback) και ανάλυση απόδοσης, ενώ συζητά τους περιορισμούς της τρέχουσας υλοποίησης.

Το Κεφάλαιο 8 (Συμπεράσματα και Μελλοντική Εργασία) συνοψίζει τα ευρήματα σε σχέση με τα ερευνητικά ερωτήματα, αξιολογεί τον βαθμό επίτευξης των στόχων και προτείνει κατευθύνσεις επέκτασης και βελτίωσης του συστήματος, τόσο σε τεχνικό όσο και σε ερευνητικό επίπεδο.

2. Θεωρητικό Υπόβαθρο

Το παρόν κεφάλαιο παρέχει τη θεωρητική τεκμηρίωση που απαιτείται για την κατανόηση της αρχιτεκτονικής που υλοποιείται στην παρούσα εργασία. Ξεκινώντας από τις θεμελιώδεις έννοιες της Επεξεργασίας Φυσικής Γλώσσας και τα Μεγάλα Γλωσσικά Μοντέλα, το κεφάλαιο προχωρά στην αρχιτεκτονική Retrieval-Augmented Generation, τις τεχνικές ευθυγράμμισης μοντέλων, και τελικά στις διανυσματικές βάσεις δεδομένων που αποτελούν την υποδομή της σημασιολογικής αναζήτησης. Κάθε ενότητα συνδέεται ρητά με τις αποφάσεις σχεδιασμού που τεκμηριώνονται στα επόμενα κεφάλαια.

2.1 Εξέλιξη της Επεξεργασίας Φυσικής Γλώσσας: Από τους Κανόνες στα Νευρωνικά Δίκτυα

Η Επεξεργασία Φυσικής Γλώσσας (NLP) ασχολείται με την αυτόματη επεξεργασία ανθρώπινης γλώσσας από υπολογιστές. Η εξέλιξη του πεδίου πέρασε από χειροποίητους κανόνες (rule-based systems) της δεκαετίας του 1950, στα στατιστικά μοντέλα της δεκαετίας του 1990, μια πρώτη μεγάλη αλλαγή παραδείγματος όπου αντί για ρητούς κανόνες τα μοντέλα (n-grams, Hidden Markov Models, Conditional Random Fields) εξήγαν γλωσσικά μοτίβα από μεγάλα σώματα κειμένου, και τελικά στα νευρωνικά δίκτυα. Κάθε αλλαγή έφερε σημαντικές βελτιώσεις σε εφαρμογές όπως η αναγνώριση ομιλίας και η ανάλυση κειμένου.

Η εισαγωγή των νευρωνικών δικτύων και της αναπαράστασης λέξεων ως διανυσμάτων (word embeddings) αποτέλεσε τη δεύτερη μεγάλη ανατροπή. Το μοντέλο Word2Vec των Mikolov et al. (2013) απέδειξε ότι η σημασιολογική σχέση λέξεων μπορεί να αναπαρασταθεί σε έναν συνεχή διανυσματικό χώρο: η διάσημη αριθμητική «Βασιλιάς − Άντρας + Γυναίκα ≈ Βασίλισσα» κατέδειξε ότι τα νευρωνικά δίκτυα μπορούν να μαθαίνουν αφηρημένες σχέσεις μόνο από εμπειρία με δεδομένα, χωρίς ανθρώπινη παρέμβαση.

Το 2018 εισήχθηκαν των προ-εκπαιδευμένων γλωσσικών μοντέλων: τα BERT (Devlin et al., 2019) και GPT (Radford et al., 2018) έδειξαν ότι η προ-εκπαίδευση σε τεράστιο όγκο κειμένου, ακολουθούμενη από εξειδίκευση (fine-tuning) σε ειδικές εργασίες, μπορεί να αποδώσει αποτελέσματα υπεράνθρωπης ακρίβειας σε πολλά NLP benchmarks. Ο δρόμος για τα σύγχρονα LLMs είχε ανοίξει.

2.2 Μεγάλα Γλωσσικά Μοντέλα (Large Language Models, LLMs)

2.2.1 Η Αρχιτεκτονική Transformer

Η αρχιτεκτονική Transformer (Vaswani et al., 2017) αποτελεί τη βάση όλων των σύγχρονων LLMs. Αντί να επεξεργάζεται κείμενο λέξη-λέξη σε σειρά (όπως τα παλαιότερα RNNs), επεξεργάζεται ολόκληρη την ακολουθία ταυτόχρονα μέσω ενός μηχανισμού που ονομάζεται self-attention.

Ο μηχανισμός αυτο-προσοχής επιτρέπει σε κάθε λέξη να «κοιτάζει» και να σταθμίζει τη σημασία όλων των άλλων λέξεων στην πρόταση, ανεξάρτητα από την απόστασή τους. Για παράδειγμα, στην πρόταση «Το κατάστημα έστειλε το δέμα αλλά αυτό καταστράφηκε κατά τη μεταφορά», ο Transformer μπορεί να κατανοήσει ότι η αντωνυμία «αυτό» αναφέρεται στο «δέμα» και όχι στο «κατάστημα», κάτι που τα προηγούμενα μοντέλα αδυνατούσαν να αντιμετωπίσουν αξιόπιστα.

Τεχνικά, ο μηχανισμός αυτο-προσοχής υπολογίζει για κάθε ζεύγος θέσεων (i, j) στην ακολουθία έναν βαθμό σημαντικότητας, τον οποίο χρησιμοποιεί για να ζυγίσει τη συνεισφορά κάθε λέξης στην αναπαράσταση των άλλων. Η διαδικασία αυτή επαναλαμβάνεται παράλληλα σε πολλαπλές «κεφαλές προσοχής» (multi-head attention), επιτρέποντας στο μοντέλο να μαθαίνει διαφορετικούς τύπους σχέσεων ταυτόχρονα (συντακτικές, σημασιολογικές, αναφορικές).

Μια κρίσιμη πρόκληση στους Transformers είναι ότι δεν έχουν εγγενή αίσθηση της σειράς των λέξεων, η επεξεργασία είναι παράλληλη. Για αυτό χρησιμοποιούνται positional encodings: μαθηματικές αναπαραστάσεις που προστίθενται στα embeddings των λέξεων για να

κωδικοποιήσουν τη θέση τους στην ακολουθία. Με αυτόν τον τρόπο, «η Αθήνα βρίσκεται στην Ελλάδα» και «η Ελλάδα βρίσκεται στην Αθήνα» αντιμετωπίζονται διαφορετικά, παρότι αποτελούνται από τις ίδιες λέξεις.

2.2.2 Προ-εκπαίδευση και Κλίμακα

Τα σύγχρονα LLMs εκπαιδεύονται αρχικά σε τεράστιο όγκο κειμένου (pre-training), όπου το μοντέλο μαθαίνει να προβλέπει την επόμενη λέξη της ακολουθίας. Παρότι ο στόχος εκπαίδευσης ακούγεται απλός, το αποτέλεσμα είναι μοντέλα που αναπτύσσουν πλούσιες εσωτερικές αναπαραστάσεις της γλώσσας και των σχέσεων μεταξύ εννοιών.

Η κλίμακα (scale) αποδείχθηκε κρίσιμος παράγοντας: οι Kaplan et al. (2020) τεκμηρίωσαν τους «νόμους κλιμάκωσης» (scaling laws), σύμφωνα με τους οποίους η απόδοση των LLMs βελτιώνεται αξιόπιστα και προβλέψιμα καθώς αυξάνονται οι παράμετροι, τα δεδομένα και οι υπολογιστικοί πόροι. Πέρα από τη σταδιακή βελτίωση, σε μεγάλη κλίμακα παρατηρούνται και «αναδυόμενες ικανότητες» (emergent abilities), δηλαδή δεξιότητες που εμφανίζονται μόνο όταν το μοντέλο υπερβεί ένα κρίσιμο μέγεθος (Wei et al., 2022).

2.2.3 Δομικά Προβλήματα των LLMs σε Επιχειρησιακό Πλαίσιο

Παρά τις εντυπωσιακές γενικές τους ικανότητες, τα LLMs εμφανίζουν τρεις δομικές αδυναμίες όταν χρησιμοποιούνται σε επιχειρησιακό πλαίσιο: ψευδαισθήσεις, γνωσιακή απαξίωση και έλλειψη πρόσβασης σε ιδιωτικά δεδομένα. Καθεμία από αυτές αναλύεται παρακάτω.

Το σημαντικότερο πρόβλημα είναι οι ψευδαισθήσεις (hallucinations): η τάση του μοντέλου να παράγει πληροφορίες που ακούγονται πειστικές αλλά είναι εσφαλμένες. Σε ένα σύστημα εξυπηρέτησης πελατών, αυτό μπορεί να εκδηλωθεί ως λανθασμένη αναφορά της πολιτικής επιστροφών του καταστήματος, εφευρεμένοι χρόνοι παράδοσης ή ανύπαρκτες προσφορές. Το φαινόμενο οφείλεται στο γεγονός ότι το μοντέλο δεν «γνωρίζει» με την ανθρώπινη έννοια, παράγει την πιο πιθανή επόμενη λέξη βάσει στατιστικών μοτίβων, ακόμα και όταν αυτό οδηγεί σε εσφαλμένες δηλώσεις.

Το δεύτερο πρόβλημα είναι η γνωσιακή απαξίωση (knowledge cutoff): τα LLMs εκπαιδεύονται σε δεδομένα μέχρι μια ορισμένη ημερομηνία και δεν έχουν πρόσβαση σε μεταγενέστερες πληροφορίες. Για μια επιχείρηση όπου οι τιμές, τα προϊόντα και οι πολιτικές αλλάζουν συχνά, αυτό σημαίνει ότι ένα LLM χωρίς πρόσθετη υποδομή δεν μπορεί ποτέ να είναι ενημερωμένο.

Το τρίτο πρόβλημα είναι ότι τα LLMs δεν έχουν εγγενή πρόσβαση σε ιδιωτικά δεδομένα επιχειρήσεων: δεν γνωρίζουν τα προϊόντα ενός συγκεκριμένου καταστήματος, την πολιτική επιστροφών του, ή τις ειδικές προσφορές. Η αντιμετώπιση αυτού με απευθείας ενσωμάτωση

ολόκληρης της γνωσιακής βάσης στο system prompt δεν είναι πρακτικά εφαρμόσιμη για μεγάλες βάσεις γνώσης, λόγω περιορισμών στο context window και αύξησης κόστους.

2.3 Αναπαράσταση Κειμένου ως Διανύσματα: Embeddings και Σημασιολογική Ομοιότητα

Για να μπορέσει ένα σύστημα RAG να εντοπίσει ποια έγγραφα της γνωσιακής βάσης σχετίζονται με μια ερώτηση, χρειάζεται έναν τρόπο να αναπαραστήσει το νόημα του κειμένου σε αριθμητική μορφή. Η παρούσα ενότητα εισάγει την έννοια των embeddings, δηλαδή της αναπαράστασης κειμένου ως διανύσματος σε πολυδιάστατο χώρο, και στη συνέχεια εξηγεί την ομοιότητα συνημιτόνου (cosine similarity), τη μετρική με την οποία το σύστημα ποσοτικοποιεί πόσο κοντά βρίσκονται σημασιολογικά δύο κείμενα.

2.3.1 Τι είναι τα Embeddings;

Ένα embedding είναι η αριθμητική αναπαράσταση ενός κειμένου (λέξης, πρότασης ή παραγράφου) ως διάνυσμα σε έναν πολυδιάστατο χώρο. Το κρίσιμο χαρακτηριστικό των embeddings είναι ότι η γεωμετρική απόσταση στον διανυσματικό χώρο αντιστοιχεί σε σημασιολογική ομοιότητα: κείμενα με παρόμοιο νόημα βρίσκονται κοντά στον χώρο, ενώ κείμενα με διαφορετικό νόημα βρίσκονται μακριά.

Για να γίνει αυτό διαισθητικό: σε έναν απλοποιημένο δισδιάστατο χώρο, η φράση «θέλω να επιστρέψω ένα προϊόν» και η φράση «πώς κάνω επιστροφή αγοράς» θα βρίσκονταν πολύ κοντά, παρότι δεν μοιράζονται κοινές λέξεις. Αντίθετα, η φράση «ποιες είναι οι μεθόδοι πληρωμής» θα βρισκόταν μακρύτερα και από τις δύο. Αυτή η ιδιότητα, που ονομάζεται σημασιολογική αναζήτηση (semantic search), είναι θεμελιώδης για τη λειτουργία της αρχιτεκτονικής RAG.

Στην παρούσα εργασία χρησιμοποιείται το μοντέλο paraphrase-multilingual-MiniLM-L12-v2, το οποίο παράγει διανύσματα 384 διαστάσεων. Η επιλογή του συγκεκριμένου μοντέλου αιτιολογείται αναλυτικά στο Κεφάλαιο 4, αλλά ένα κρίσιμο χαρακτηριστικό του αξίζει να αναφερθεί εδώ: η πολυγλωσσική φύση του επιτρέπει τη διαγλωσσική σημασιολογική αντιστοίχιση (cross-lingual semantic matching). Αυτό σημαίνει ότι μια ερώτηση στα αγγλικά («what is your return policy?») μπορεί να αντιστοιχιστεί σωστά με ένα έγγραφο στα ελληνικά («πολιτική επιστροφών, 30 ημέρες»), χωρίς ρητή μετάφραση (Reimers & Gurevych, 2020).

2.3.2 Ομοιότητα Συνημιτόνου (Cosine Similarity)

Η μετρική που χρησιμοποιείται για τη σύγκριση δύο διανυσμάτων embeddings είναι η ομοιότητα συνημιτόνου (cosine similarity). Αντί να μετρά την ευκλείδεια απόσταση μεταξύ δύο διανυσμάτων (που εξαρτάται από το μέγεθός τους), υπολογίζει τη γωνία μεταξύ τους, μια μέτρηση που είναι

ανεξάρτητη από το μέγεθος και εστιάζει στην κατεύθυνση του διανύσματος, η οποία αντιστοιχεί στο νόημα.

Η μαθηματική της έκφραση για δύο διανύσματα A και B ορίζεται ως:

cosine_similarity(A, B) = (A · B) / (||A|| × ||B||)

Το αποτέλεσμα κυμαίνεται από −1 έως +1, όπου +1 σημαίνει ταυτόσημη κατεύθυνση (πανομοιότυπο νόημα), 0 σημαίνει κάθετα διανύσματα (ασχετο νόημα), και −1 σημαίνει αντίθετη κατεύθυνση (αντίθετο νόημα). Στην πράξη, για σύγκριση θετικών κειμένων, οι τιμές κυμαίνονται συνήθως μεταξύ 0 και 1.

Στο σύστημα που αναπτύσσεται, χρησιμοποιείται ένα κατώφλι ομοιότητας (similarity threshold) τιμής 0,5: μόνο έγγραφα με ομοιότητα ≥ 0,5 θεωρούνται αρκετά σχετικά για να συμπεριληφθούν στο context του LLM. Αυτό αποτρέπει την έγχυση άσχετων πληροφοριών που θα μπορούσαν να μπερδέψουν ή να παραπλανήσουν το μοντέλο.

2.4 Αρχιτεκτονική Retrieval-Augmented Generation (RAG)

Έχοντας περιγράψει πώς αναπαρίσταται το νόημα του κειμένου ως διανύσματα, η ενότητα αυτή παρουσιάζει την αρχιτεκτονική Retrieval-Augmented Generation (RAG), που αποτελεί τον πυρήνα του συστήματος. Αρχικά δίνεται ο ορισμός και μια σύντομη ιστορική αναδρομή της αρχιτεκτονικής, έπειτα αναλύονται τα στάδια του RAG pipeline (offline και online φάση), και τέλος γίνεται συγκριτική ανάλυση της RAG έναντι της μικρορύθμισης (fine-tuning), τεκμηριώνοντας γιατί η RAG είναι καταλληλότερη για ένα σύστημα με δυναμικά επιχειρησιακά δεδομένα.

2.4.1 Ορισμός και Ιστορική Αναδρομή

Η αρχιτεκτονική Retrieval-Augmented Generation (RAG) εισήχθη από τους Lewis et al. (2020) ως λύση στο πρόβλημα της γνωσιακής απαξίωσης και των ψευδαισθήσεων στα LLMs. Η βασική ιδέα συνδυάζει δύο υπαρκτές παραδόσεις στην τεχνητή νοημοσύνη: τα γλωσσικά μοντέλα (για παραγωγή φυσικού λόγου) με την ανάκτηση πληροφοριών (Information Retrieval, IR, για εντοπισμό σχετικών εγγράφων).

Μια χρήσιμη αναλογία: αντί το bot να απαντά αποκλειστικά από μνήμη, του δίνεται η δυνατότητα να συμβουλευτεί τη γνωσιακή βάση της επιχείρησης πριν απαντήσει. Η ποιότητα της απάντησης εξαρτάται τότε από το περιεχόμενο της βάσης, την οποία μπορεί κάποιος να ενημερώνει χωρίς να επανεκπαιδεύει το μοντέλο.

Από τη δημοσίευσή της, η αρχιτεκτονική RAG έχει εξελιχθεί σημαντικά. Αρχικά, η ανάκτηση γινόταν αποκλειστικά με παραδοσιακές τεχνικές αντιστοίχισης λέξεων-κλειδιών (sparse retrieval, BM25). Η εισαγωγή των dense embeddings από νευρωνικά δίκτυα επέτρεψε τη σημασιολογική αναζήτηση (dense retrieval), η οποία χρησιμοποιείται στην παρούσα υλοποίηση. Σύγχρονες παραλλαγές (Hybrid RAG, Agentic RAG) συνδυάζουν και τις δύο τεχνικές ή επεκτείνουν το RAG με εκτελέσιμες ενέργειες (Gao et al., 2024).

2.4.2 Τα Στάδια του RAG Pipeline

Το RAG pipeline αποτελείται από δύο κύριες φάσεις: την Offline φάση (προεπεξεργασία) και την Online φάση (χρόνος εκτέλεσης).

Κατά την Offline φάση, κάθε έγγραφο της γνωσιακής βάσης (FAQ, άρθρο, εγχειρίδιο) μετατρέπεται σε διάνυσμα (embedding) και αποθηκεύεται σε μια διανυσματική βάση δεδομένων. Αυτή η φάση εκτελείται μία φορά (ή κάθε φορά που ενημερώνεται η γνωσιακή βάση) και δεν επηρεάζει τον χρόνο απόκρισης. Στην παρούσα υλοποίηση, αυτό γίνεται κατά το seeding της βάσης και κατά κάθε προσθήκη ή επεξεργασία εγγραφής μέσω του admin panel.

Κατά την Online φάση, η ερώτηση του χρήστη μετατρέπεται σε διάνυσμα (query embedding), εντοπίζονται τα N πιο σχετικά έγγραφα από τη βάση (top-N retrieval) βάσει ομοιότητας συνημιτόνου, τα έγγραφα αυτά εισάγονται ως context στο σύστημα οδηγιών (system prompt) του LLM, και τέλος το LLM παράγει απάντηση που στηρίζεται αποκλειστικά στο παρεχόμενο context.

Ένα κρίσιμο χαρακτηριστικό της Online φάσης είναι ο χειρισμός της έλλειψης σχετικών εγγράφων. Όταν η ερώτηση είναι εκτός της γνωσιακής βάσης (π.χ. «ποια η καλύτερη συνταγή για μουσακά;»), το RAG δεν ανακτά κανένα σχετικό έγγραφο (ομοιότητα < 0,5 για όλα τα έγγραφα). Το σύστημα οδηγιών στη συνέχεια εντέλλει το LLM να απαντήσει ευγενικά ότι δεν μπορεί να βοηθήσει με το συγκεκριμένο θέμα, αποτρέποντας τις ψευδαισθήσεις χωρίς αγένεια.

2.4.3 RAG έναντι Fine-Tuning: Συγκριτική Ανάλυση

Μια κεντρική ερώτηση στη βιβλιογραφία είναι η επιλογή μεταξύ RAG και μικρορύθμισης (fine-tuning) για την ενσωμάτωση ειδικής γνώσης σε LLMs. Οι δύο προσεγγίσεις διαφέρουν ριζικά στον τρόπο αποθήκευσης της γνώσης και τις αντίστοιχες εκπαιδευτικές, λειτουργικές και κόστους επιπτώσεις.

Στο fine-tuning, η νέα γνώση ενσωματώνεται απευθείας στις παραμέτρους του μοντέλου μέσω συνέχισης της εκπαίδευσης σε εξειδικευμένο corpus. Το αποτέλεσμα είναι ένα μοντέλο που «ξέρει» τις πληροφορίες χωρίς να χρειάζεται εξωτερική βάση δεδομένων. Ωστόσο, το fine-tuning απαιτεί σημαντικούς υπολογιστικούς πόρους (GPU ώρες), μεγάλο και καλά δομημένο dataset εκπαίδευσης, και επαναλαμβάνεται κάθε φορά που αλλάζει η γνώση. Για επιχειρήσεις με δυναμικά δεδομένα

(αλλαγές τιμών, νέα προϊόντα, αναθεωρήσεις πολιτικής), αυτό καθιστά το fine-tuning εξαιρετικά αδύναμο ή και απρακτικό.

Στη RAG, η γνώση παραμένει εξωτερική, σε ανθρωπίνως αναγνώσιμη και επεξεργάσιμη μορφή. Η ενημέρωση της γνωσιακής βάσης (π.χ. αλλαγή πολιτικής επιστροφών από 30 σε 14 ημέρες) απαιτεί μόνο ενημέρωση της εγγραφής στη βάση δεδομένων και επαναδιανυσματοποίηση, διαδικασία δευτερολέπτων, χωρίς εκπαίδευση. Η ιχνηλασιμότητα είναι επίσης σημαντικό πλεονέκτημα: για κάθε απάντηση μπορεί να εντοπιστεί ακριβώς ποιο έγγραφο χρησιμοποιήθηκε ως πηγή.

Οι δύο μέθοδοι δεν είναι αμοιβαία αποκλειστικές, καθώς πολλά production συστήματα τις συνδυάζουν, ενώ η αιτιολόγηση της επιλογής της RAG για την παρούσα υλοποίηση αναλύεται στο Κεφάλαιο 4.

2.5 Ευθυγράμμιση Μοντέλων: RLHF και Human-in-the-Loop

Η ανάκτηση σχετικής πληροφορίας δεν αρκεί από μόνη της για να καταστήσει ένα γλωσσικό μοντέλο χρήσιμο και αξιόπιστο· απαιτείται επιπλέον η ευθυγράμμισή του με τις ανθρώπινες προθέσεις. Η ενότητα αυτή εξετάζει το πρόβλημα της ευθυγράμμισης (alignment problem), παρουσιάζει την τεχνική Reinforcement Learning from Human Feedback (RLHF) που χρησιμοποιείται για την ευθυγράμμιση των σύγχρονων LLMs, και καταλήγει στην προσέγγιση της ευθυγράμμισης κατά τον χρόνο εκτέλεσης (inference-time alignment) που υιοθετεί η παρούσα εργασία μέσω του μηχανισμού Human-in-the-Loop.

2.5.1 Το Πρόβλημα της Ευθυγράμμισης (Alignment Problem)

Ένα LLM που εκπαιδεύτηκε μόνο με next-token prediction δεν είναι αυτόματα χρήσιμο ή ασφαλές. Απλώς παράγει στατιστικά πιθανό κείμενο. Η ευθυγράμμιση (alignment) είναι η διαδικασία μέσω της οποίας το μοντέλο προσαρμόζεται ώστε να ακολουθεί οδηγίες, να αποφεύγει επιβλαβείς απαντήσεις και να παραδέχεται αβεβαιότητα.

Η ευθυγράμμιση έχει τρεις διαστάσεις που συχνά αναφέρονται ως «HHH»: Helpful (χρήσιμο, να απαντά στις ερωτήσεις αποτελεσματικά), Harmless (ακίνδυνο, να αποφεύγει επιβλαβείς απαντήσεις), και Honest (ειλικρινές, να παραδέχεται αβεβαιότητα αντί να εφευρίσκει). Η επίτευξη και των τριών ταυτόχρονα αποτελεί ενεργό ερευνητικό πεδίο.

2.5.2 Reinforcement Learning from Human Feedback (RLHF)

Η τεχνική RLHF, που τεκμηριώθηκε από τους Ouyang et al. (2022) στο InstructGPT και εφαρμόστηκε εκτεταμένα στο ChatGPT, είναι ο πιο διαδεδομένος τρόπος ευθυγράμμισης LLMs. Αποτελείται από τρεις διακριτές φάσεις.

Στη φάση Supervised Fine-Tuning (SFT), ανθρώπινοι σχολιαστές παράγουν υποδειγματικές απαντήσεις σε δείγμα ερωτήσεων, και το μοντέλο εκπαιδεύεται να μιμείται αυτές τις απαντήσεις. Αυτό δίνει στο μοντέλο αρχική κατεύθυνση για τον τρόπο που πρέπει να απαντά.

Στη φάση Reward Modeling, ανθρώπινοι αξιολογητές κατατάσσουν ζεύγη απαντήσεων από το SFT μοντέλο (π.χ. «ποια από αυτές τις δύο απαντήσεις είναι καλύτερη;»). Αυτές οι κατατάξεις χρησιμοποιούνται για να εκπαιδευτεί ένα ξεχωριστό νευρωνικό δίκτυο, το Reward Model, που μαθαίνει να αποδίδει υψηλές βαθμολογίες σε απαντήσεις που οι άνθρωποι προτιμούν.

Στη φάση RL Optimization, το κύριο γλωσσικό μοντέλο βελτιστοποιείται μέσω αλγορίθμων ενισχυτικής μάθησης (συνήθως PPO, Proximal Policy Optimization) για να μεγιστοποιεί τη βαθμολογία του Reward Model, ενώ ταυτόχρονα διατηρεί εγγύτητα με το αρχικό SFT μοντέλο (για να αποφεύγεται η «κατάρρευση» της γλωσσικής ικανότητας). (Schulman et al., 2017)

Το πλήρες RLHF pipeline είναι εξαιρετικά υπολογιστικά ακριβό και απαιτεί εκατοντάδες χιλιάδες ανθρώπινες αξιολογήσεις, GPU cluster για εκπαίδευση, και εξειδικευμένη τεχνογνωσία. Για αυτό δεν είναι εφαρμόσιμο στα πλαίσια της παρούσας εργασίας, ωστόσο, ο μηχανισμός Human-in-the-Loop που υλοποιείται αποτελεί προσομοίωση της φάσης Reward Modeling σε επίπεδο inference time.

2.5.3 Inference-Time Alignment: Η Προσέγγιση της Παρούσας Εργασίας

Στη βιβλιογραφία αναφέρεται ολοένα συχνότερα η έννοια του ευθυγράμμιση κατά την εκτέλεση: η τροποποίηση της συμπεριφοράς ενός LLM κατά τον χρόνο εκτέλεσης, χωρίς αλλαγή των παραμέτρων του. Η πιο κοινή τεχνική είναι η few-shot prompting: η ενσωμάτωση ζευγαριών ερώτησης-απάντησης στο system prompt ως «παραδείγματα» που καθοδηγούν τη συμπεριφορά του μοντέλου.

Στην παρούσα εργασία, αυτή η τεχνική συνδυάζεται με έναν Human-in-the-Loop μηχανισμό: όταν ένας χρήστης βαθμολογεί αρνητικά μια απάντηση και προτείνει διόρθωση, ένας διαχειριστής αξιολογεί και εγκρίνει τη διόρθωση μέσω του admin panel. Η εγκεκριμένη διόρθωση αποθηκεύεται ως «χρυσός κανόνας» (golden rule) και εισάγεται αυτόματα σε κάθε επόμενο system prompt ως few-shot παράδειγμα.

Αυτή η προσέγγιση αντιστοιχεί λειτουργικά στη φάση Reward Modeling του RLHF: οι ανθρώπινες κατατάξεις («αυτή η απάντηση είναι λάθος, η σωστή είναι αυτή») χρησιμοποιούνται για να βελτιώσουν τη συμπεριφορά του συστήματος. Η διαφορά είναι ότι αντί να επαναεκπαιδεύεται το μοντέλο, οι βελτιώσεις εφαρμόζονται άμεσα μέσω του context, αποτελεσματικότερη για μικρά

datasets, αλλά με περιορισμό στον αριθμό των κανόνων που μπορούν να εισαχθούν χωρίς να υπερφορτωθεί το context window.

2.6 Διανυσματικές Βάσεις Δεδομένων και MongoDB Atlas Vector Search

Η πρακτική υλοποίηση της σημασιολογικής αναζήτησης προϋποθέτει μια υποδομή ικανή να αποθηκεύει και να αναζητά αποδοτικά διανύσματα υψηλής διαστατικότητας. Η ενότητα αυτή εξηγεί γιατί οι παραδοσιακές βάσεις δεδομένων δεν επαρκούν για τον σκοπό αυτό και πώς οι διανυσματικές βάσεις τον αντιμετωπίζουν, παρουσιάζει τη λειτουργία $vectorSearch της MongoDB Atlas που χρησιμοποιείται στην υλοποίηση, και αναδεικνύει τα πλεονεκτήματα της ενοποιημένης αποθήκευσης δεδομένων και διανυσμάτων σε ένα ενιαίο σύστημα.

2.6.1 Η Ανάγκη για Εξειδικευμένη Αποθήκευση Διανυσμάτων

Η λειτουργία του RAG απαιτεί την αποτελεσματική αποθήκευση και αναζήτηση εκατομμυρίων διανυσμάτων υψηλής διαστατικότητας. Οι παραδοσιακές σχεσιακές βάσεις δεδομένων δεν είναι σχεδιασμένες για αυτή τη χρήση: μια naïve προσέγγιση που συγκρίνει κάθε ερώτημα με κάθε αποθηκευμένο διάνυσμα (brute-force search) έχει πολυπλοκότητα O(n·d), όπου n είναι ο αριθμός των εγγράφων και d η διαστατικότητα των διανυσμάτων. Για μεγάλες βάσεις γνώσης, αυτό καθίσταται μη αποδεκτό.

Οι διανυσματικές βάσεις δεδομένων (vector databases) επιλύουν αυτό το πρόβλημα χρησιμοποιώντας ειδικευμένες δομές ευρετηρίασης (indexing structures) που επιτρέπουν προσεγγιστική αναζήτηση πλησιέστερων γειτόνων (Approximate Nearest Neighbor, ANN) με πολύ χαμηλότερη πολυπλοκότητα. Μια από τις πιο ευρέως χρησιμοποιούμενες δομές είναι το HNSW (Hierarchical Navigable Small World graph), το οποίο χρησιμοποιεί και το MongoDB Atlas (Malkov & Yashunin, 2020).

2.6.2 MongoDB Atlas $vectorSearch

Η MongoDB, παραδοσιακά γνωστή ως document-oriented βάση δεδομένων, εισήγαγε τη λειτουργία $vectorSearch στην Atlas πλατφόρμα, επιτρέποντας την αποθήκευση και σημασιολογική αναζήτηση διανυσμάτων εντός της ίδιας βάσης που αποθηκεύονται και τα υπόλοιπα δεδομένα (MongoDB, 2024). Αυτό εξαλείφει την ανάγκη για ξεχωριστή διανυσματική βάση (π.χ. Pinecone, Milvus, Chroma), μειώνοντας δραματικά την πολυπλοκότητα της αρχιτεκτονικής.

Για την ενεργοποίηση του $vectorSearch, απαιτείται η δημιουργία ενός Atlas Search Index με τύπο «vectorSearch» στη συλλογή που περιέχει τα embeddings. Ο index ορίζει το πεδίο που περιέχει τα διανύσματα, τον αριθμό διαστάσεων (384 στην παρούσα υλοποίηση) και τη μετρική ομοιότητας (cosine). Μόλις δημιουργηθεί, η MongoDB χτίζει και διατηρεί αυτόματα τον HNSW index.

Η αναζήτηση εκτελείται ως MongoDB Aggregation Pipeline με το στάδιο $vectorSearch, το οποίο δέχεται το διάνυσμα του ερωτήματος (queryVector), τον αριθμό υποψηφίων που θα εξεταστούν (numCandidates, για HNSW ANN search), και τον αριθμό αποτελεσμάτων που θα επιστραφούν (limit). Ακολουθεί ένα στάδιο $addFields που προσθέτει τη βαθμολογία ομοιότητας, και ένα $match που φιλτράρει αποτελέσματα κάτω από το κατώφλι.

2.6.3 Πλεονεκτήματα της Ενοποιημένης Προσέγγισης

Η επιλογή της MongoDB ως ενιαίου αποθετηρίου για συνομιλίες, ανατροφοδοτήσεις και διανύσματα παρέχει αρκετά πλεονεκτήματα. Καταρχάς απλοποιεί την υποδομή, καθώς δεν απαιτείται συγχρονισμός δεδομένων μεταξύ διαφορετικών συστημάτων, κάτι που μειώνει τον κίνδυνο ασυμφωνίας δεδομένων (data drift). Ταυτόχρονα ελαχιστοποιεί τον λανθάνοντα χρόνο, αφού οι αναζητήσεις RAG και η ανάκτηση ιστορικού συνομιλίας γίνονται στο ίδιο σύστημα, χωρίς δικτυακή επικοινωνία μεταξύ υπηρεσιών. Τέλος, είναι οικονομικά αποδοτική: η MongoDB Atlas M0 (Free Tier) επαρκεί για τις ανάγκες της παρούσας πτυχιακής, με μηδενικό κόστος λειτουργίας.

Ένα πρακτικό θέμα που αντιμετωπίζεται είναι η συμβατότητα $vectorSearch αποκλειστικά με το MongoDB Atlas, δεν λειτουργεί σε τοπικές εγκαταστάσεις MongoDB. Για το λόγο αυτό, η υλοποίηση περιλαμβάνει αυτόματη ανίχνευση του περιβάλλοντος: εάν η συμβολοσειρά σύνδεσης ξεκινά με mongodb+srv:// (Atlas), χρησιμοποιείται $vectorSearch, ενώ για τοπικές εγκαταστάσεις χρησιμοποιείται $text search ως εναλλακτική λύση.

2.7 Συνοψίζοντας: Η Σύνθεση των Τεχνολογιών

Τα τέσσερα θεωρητικά πεδία που αναλύθηκαν στο παρόν κεφάλαιο, LLMs, embeddings, RAG και RLHF/HitL, ενσωματώνονται σε ένα ενιαίο, συμπληρωματικό σύστημα. Το LLM παρέχει τη γλωσσική ικανότητα, τα embeddings επιτρέπουν τη σημασιολογική αντιστοίχιση ερώτησης-απάντησης, το RAG «αγκυρώνει» το LLM στην αξιόπιστη επιχειρησιακή γνώση, και ο HitL μηχανισμός επιτρέπει τη συνεχή βελτίωση της συμπεριφοράς του συστήματος χωρίς επανεκπαίδευση.

Συνοψίζοντας, κάθε τεχνολογία που αναλύθηκε απαντά σε μία ξεχωριστή αρχιτεκτονική ανάγκη: τα LLMs παρέχουν την ικανότητα παραγωγής φυσικής γλώσσας, τα embeddings επιτρέπουν τη σημασιολογική σύγκριση κειμένων, η RAG αρχιτεκτονική ενσωματώνει εξωτερική γνώση χωρίς επανεκπαίδευση, και ο μηχανισμός HitL επιτρέπει τη βελτίωση του συστήματος από ανθρώπινες διορθώσεις σε πραγματικό χρόνο.

3. Ανάλυση Αγοράς και Ανταγωνιστικό Τοπίο

Το παρόν κεφάλαιο εξετάζει το τρέχον τοπίο των λύσεων τεχνητής νοημοσύνης στην εξυπηρέτηση πελατών, με διπλό στόχο. Αφενός επιδιώκει να τεκμηριώσει την αγοραία πραγματικότητα μέσα στην οποία τοποθετείται η παρούσα εργασία, με αξιόπιστα στατιστικά στοιχεία που αναδεικνύουν τη σημασία του πεδίου. Αφετέρου αναλύει κριτικά τις κυρίαρχες εμπορικές πλατφόρμες, εντοπίζοντας τα δομικά κενά που δικαιολογούν την ανάπτυξη μιας custom, ανοιχτής αρχιτεκτονικής. Το κεφάλαιο καταλήγει σε μια σύγκριση που τοποθετεί ρητά την παρούσα υλοποίηση έναντι των εμπορικών εναλλακτικών.

3.1 Το Μέγεθος και η Δυναμική της Αγοράς

3.1.1 Παγκόσμιοι Αριθμοί

Η αγορά chatbots και εικονικών βοηθών αναπτύσσεται με γρήγορους ρυθμούς. Σύμφωνα με την Grand View Research (2025), η παγκόσμια αγορά conversational AI αποτιμήθηκε σε περίπου 11,58 δισεκατομμύρια δολάρια το 2024 και αναμένεται να φτάσει τα 41,39 δισεκατομμύρια έως το 2030 (CAGR 23,7%). Ο κλάδος του ηλεκτρονικού εμπορίου είναι ένας από τους κύριους κινητήριους μοχλούς.

Από την πλευρά της υιοθέτησης, το AI Index του Stanford Institute for Human-Centered AI (2025) αναφέρει ότι το 78% των οργανισμών χρησιμοποίησε τεχνητή νοημοσύνη σε τουλάχιστον μία επιχειρηματική λειτουργία το 2024, από 55% μόλις έναν χρόνο νωρίτερα. Η ταχεία αυτή διάδοση, ωστόσο, δεν συνεπάγεται και ώριμη ενσωμάτωση: το χάσμα μεταξύ τεχνολογικής δυνατότητας και αποτελεσματικής πρακτικής εφαρμογής παραμένει σημαντικό, καθώς η τεχνολογία υιοθετείται ταχύτερα από όσο εδραιώνονται οι αποδοτικές πρακτικές χρήσης της.

Τα οφέλη για τις επιχειρήσεις που υιοθετούν επιτυχώς λύσεις AI εξυπηρέτησης πελατών είναι τεκμηριωμένα: σε προηγμένα μοντέλα ψηφιακής εξυπηρέτησης, τα κανάλια αυτοεξυπηρέτησης μπορούν να διαχειριστούν το 70–80% των αλληλεπιδράσεων (McKinsey & Company, 2022), με σημαντική μείωση του λειτουργικού κόστους και διαθεσιμότητα 24/7 χωρίς πρόσθετο κόστος υπερωριών. Η τάση αυτή εντείνεται: σύμφωνα με τη Salesforce (2025), έως το 2027 η τεχνητή νοημοσύνη αναμένεται να επιλύει το 50% των περιπτώσεων εξυπηρέτησης, από 30% το 2025.

3.1.2 Αποτυχίες και Ανεκπλήρωτες Προσδοκίες

Παρά τα θετικά στοιχεία, τα ποσοστά αποτυχίας παραμένουν υψηλά. Σημαντικό μέρος των καταναλωτών αναφέρει ότι τα chatbots συχνά δεν καταφέρνουν να λύσουν το πρόβλημά τους, ενώ μεγάλη μερίδα εκφράζει επιφυλακτική ή αρνητική στάση απέναντι στην υπερβολική εξάρτηση από AI στην εξυπηρέτηση (Castillo & Farrugia Caruana, 2025). Αυτά τα ευρήματα δεν αντιφάσκουν απαραίτητα: οι επιχειρήσεις επενδύουν, αλλά τα συστήματα συχνά δεν ανταποκρίνονται στις προσδοκίες.

Τα κύρια αίτια αποτυχίας εντοπίζονται σε τρία επίπεδα. Το πιο κρίσιμο είναι ότι οι ψευδαισθήσεις και οι αναξιόπιστες απαντήσεις διαβρώνουν την εμπιστοσύνη: ένα chatbot που δίνει λανθασμένη πληροφορία για πολιτική επιστροφών δεν απλώς αποτυγχάνει, δημιουργεί νομική έκθεση για την επιχείρηση. Σε ένα δεύτερο επίπεδο, η αδυναμία χειρισμού πολύπλοκων ή συναισθηματικά φορτισμένων ερωτημάτων οδηγεί σε απογοήτευση χρηστών που νιώθουν ότι «μιλούν με μηχανή». Υπάρχει τέλος η αδυναμία βελτίωσης χωρίς επανεκπαίδευση, που σημαίνει ότι τα ίδια λάθη επαναλαμβάνονται χωρίς τρόπο διόρθωσης.

Αυτό το πλαίσιο καθιστά σαφές γιατί δεν αρκεί να «αναπτύξει» κανείς απλώς ένα LLM σε ρόλο εξυπηρέτησης πελατών: απαιτείται αρχιτεκτονική που αντιμετωπίζει ρητά τα παραπάνω δομικά προβλήματα. Αυτή είναι η αφετηρία για την αξιολόγηση των εμπορικών λύσεων που ακολουθεί.

3.2 Κυρίαρχες Εμπορικές Πλατφόρμες: Αναλυτική Κριτική

Η αγορά εμπορικών λύσεων AI εξυπηρέτησης πελατών χαρακτηρίζεται από λίγους, ισχυρούς παίκτες με διαφορετικές αρχιτεκτονικές προσεγγίσεις και μοντέλα τιμολόγησης. Στις επόμενες υποενότητες αναλύονται κριτικά οι πέντε σημαντικότερες πλατφόρμες, με έμφαση στις αρχιτεκτονικές επιλογές, τα πλεονεκτήματα και τα δομικά μειονεκτήματά τους.

Συγκριτική Επισκόπηση: Οι κυριότερες εμπορικές πλατφόρμες που εξετάστηκαν (Intercom Fin, Zendesk AI, Salesforce Einstein/Agentforce, Freshdesk Freddy AI και Tidio Lyro) καλύπτουν ένα ευρύ φάσμα από mid-market έως enterprise λύσεις. Το Intercom Fin διακρίνεται για τη γρήγορη παραγωγική εκκίνηση αλλά με τιμολόγηση ανά "επιτυχή επίλυση" που μπορεί να γίνει απρόβλεπτο κόστος. Το Zendesk AI ενσωματώνεται βαθιά σε υπάρχουσες ροές εργασίας αλλά περιορίζεται σε rule-based λογική για πολύπλοκα σενάρια. Το Salesforce Einstein προσφέρει την πιο βαθιά CRM ενσωμάτωση αλλά με εξαιρετικά υψηλό κόστος εισόδου.

Το Freshdesk Freddy AI στοχεύει τη μέση αγορά με ισορροπημένες δυνατότητες, ενώ το Tidio Lyro αποτελεί την πιο προσιτή επιλογή για μικρές επιχειρήσεις με απλά σενάρια· η αξία τέτοιων λύσεων για πολύ μικρές επιχειρήσεις τεκμηριώνεται και σε πρόσφατες μελέτες περίπτωσης (Marcineková et al., 2025). Κοινός παρονομαστής όλων είναι ο περιορισμός στην προσαρμογή: το knowledge base και οι κανόνες ανατροφοδότησης διαχειρίζονται μέσω κλειστών UI, χωρίς πρόσβαση στον υποκείμενο μηχανισμό. Αυτό αποτελεί κεντρικό κίνητρο για την ανάπτυξη custom αρχιτεκτονικής στην παρούσα εργασία.

3.3 Συγκριτική Αξιολόγηση

Ο Πίνακας 3.1 συνοψίζει τη συγκριτική αξιολόγηση των πέντε πλατφορμών βάσει κριτηρίων που είναι κρίσιμα για την τεχνική και επιχειρησιακή αποτίμησή τους.

Πίνακας 3.1: Συγκριτική αξιολόγηση εμπορικών πλατφορμών AI εξυπηρέτησης πελατών

Κριτήριο	Intercom Fin	Zendesk AI	Salesforce	Freshdesk	Tidio Lyro
Αρχιτεκτονική	RAG + GPT-4	Fine-tuned + Rules	RAG + CRM data	ML Classification	Rule-based + FAQ
Κόστος εισόδου	Μεσαίο + $/επίλυση	Μεσαίο (add-on)	Πολύ Υψηλό	Μεσαίο	Χαμηλό
Ελληνική γλώσσα	Μερική	Περιορισμένη	Μερική	Περιορισμένη	Ελάχιστη
Έλεγχος δεδομένων	Χαμηλός	Μεσαίος	Μεσαίος	Μεσαίος	Χαμηλός
Feedback/Learning	Κλειστό	Περιορισμένο	Μέτριο	Περιορισμένο	Ελάχιστο
Προσαρμοστικότητα	Μεσαία	Μεσαία	Υψηλή	Μεσαία	Χαμηλή
Κατάλληλο για ΜΜΕ	Ναι (κόστος)	Ναι (αν ήδη Zendesk)	Όχι	Ναι	Ναι

Πηγή: Ιδία επεξεργασία βάσει τεχνικής τεκμηρίωσης παρόχων και ανεξάρτητων αναλύσεων (2024–2025).

Από τη συγκριτική ανάλυση προκύπτουν τρία κοινά χαρακτηριστικά που μοιράζονται όλες οι εμπορικές πλατφόρμες, ανεξάρτητα από το μέγεθος και την τιμή τους. Όλες λειτουργούν ως «μαύρα κουτιά»: ο πελάτης δεν έχει ορατότητα στην εσωτερική λειτουργία του μοντέλου, ούτε τη δυνατότητα να παρέμβει στη διαδικασία μάθησης. Επιπλέον, ο έλεγχος δεδομένων παραμένει μερικός, αφού τα δεδομένα συνομιλιών αποθηκεύονται στους servers της πλατφόρμας, γεγονός που εγείρει ζητήματα GDPR για ευρωπαϊκές επιχειρήσεις. Τέλος, η προσαρμογή στα συγκεκριμένα χαρακτηριστικά μιας επιχείρησης (τόνος επικοινωνίας, εξειδικευμένη ορολογία, νομικές απαιτήσεις) είναι είτε αδύνατη είτε κλειδωμένη πίσω από enterprise contracts.

3.4 Τάσεις Καταναλωτών και Προσδοκίες για το 2025

Η κατανόηση των τάσεων από την πλευρά του καταναλωτή είναι εξίσου σημαντική με την ανάλυση των τεχνικών λύσεων. Έρευνα των Castillo & Farrugia Caruana (2025) που εξέτασε τις «pre-interaction expectations», δηλαδή, τι περιμένουν οι πελάτες πριν ακόμα αρχίσουν μια συνομιλία με chatbot, αποκάλυψε σημαντικά ευρήματα.

Η ταχύτητα παραμένει ένας από τους πιο σημαντικούς παράγοντες ικανοποίησης: η πλειονότητα των καταναλωτών θεωρεί τον «σεβασμό του χρόνου τους» καθοριστικό στοιχείο της καλής εξυπηρέτησης. Τα AI συστήματα υπερτερούν εδώ έναντι ανθρώπινων agents, οι οποίοι συχνά λαμβάνουν εισερχόμενα ερωτήματα εκτός ωραρίου ή σε περιόδους αιχμής. Η ταχύτητα, ωστόσο, από μόνη της δεν αρκεί: οι χρήστες τείνουν να εγκαταλείπουν ένα chatbot όταν αυτό αποτυγχάνει να επιλύσει το πρόβλημά τους μέσα σε λίγες ανταλλαγές μηνυμάτων.

3.5 Κενά της Αγοράς και Αιτιολόγηση της Παρούσας Υλοποίησης

Από την ανάλυση που προηγήθηκε προκύπτουν τρία κενά που δεν καλύπτονται επαρκώς από τις υπάρχουσες εμπορικές λύσεις, και που αποτελούν την αιτιολόγηση για την ανάπτυξη custom αρχιτεκτονικής στην παρούσα εργασία.

Το πρώτο κενό αφορά τη διαφάνεια και τον έλεγχο. Ο κύκλος ανατροφοδότησης (feedback loop), η δυνατότητα δηλαδή να βλέπει ο διαχειριστής ποιες ερωτήσεις δεν απαντήθηκαν σωστά, να διορθώνει τις απαντήσεις, και να βλέπει αυτές τις διορθώσεις να εφαρμόζονται άμεσα, είναι ανύπαρκτος ή κλειδωμένος σε enterprise επίπεδα σε όλες τις εξεταζόμενες πλατφόρμες. Η παρούσα υλοποίηση προσφέρει αυτή τη δυνατότητα εγγενώς, χωρίς πρόσθετο κόστος.

Το δεύτερο κενό αφορά την κυριαρχία δεδομένων (κυριαρχία δεδομένων). Στο πλαίσιο του GDPR, η αποθήκευση δεδομένων συνομιλιών σε servers τρίτων (ιδιαίτερα εκτός ΕΕ) δημιουργεί νομικές υποχρεώσεις και κινδύνους. Μια self-hosted αρχιτεκτονική που εκτελείται σε υποδομή της επιχείρησης (ακόμα και σε τοπικό Docker container για μικρές επιχειρήσεις) εξαλείφει αυτή την ανησυχία εξ ορισμού.

Το τρίτο κενό αφορά το οικονομικό μοντέλο. Η χρέωση ανά επίλυση του Intercom, τα add-on costs του Zendesk, και τα enterprise contracts του Salesforce δημιουργούν απρόβλεπτη κοστολόγηση που εμποδίζει τον προϋπολογισμό. Μια αρχιτεκτονική βασισμένη σε δωρεάν εργαλεία (MongoDB Atlas M0, τοπικά embeddings, free-tier LLM APIs) προσφέρει ουσιαστικά μηδενικό λειτουργικό κόστος για ερευνητικό ή πρωτοτυπικό επίπεδο.

Πίνακας 3.2: Τοποθέτηση της παρούσας υλοποίησης έναντι εμπορικών λύσεων

Κριτήριο	Εμπορικές Λύσεις	ShopEasy ChatBot (Παρούσα Εργασία)
Κόστος λειτουργίας	€0,99/επίλυση ή συνδρομή €100–500+/μήνα	~ €0 (free tier APIs + local embeddings)
Έλεγχος δεδομένων	Περιορισμένος — servers πλατφόρμας	Πλήρης — self-hosted MongoDB
Feedback Loop	Κλειστό ή απών	Ενσωματωμένος HitL κύκλος
Ενημέρωση γνώσης	Μέσω UI πλατφόρμας	Admin panel με live embedding update
Αρχιτεκτονική διαφάνεια	Black box	Πλήρως ανοιχτός κώδικας
Πολυγλωσσική υποστήριξη	Μερική (EN κεντρική)	Cross-lingual (GR/EN) από σχεδιασμό
Εναλλαγή AI παρόχου	Αδύνατη	Μέσω ENV variable (Factory Pattern)

Πηγή: Ιδία επεξεργασία.

Ο Πίνακας 3.2 αποτυπώνει με σαφήνεια τη θέση της παρούσας υλοποίησης στο ανταγωνιστικό τοπίο. Δεν επιδιώκει να ανταγωνιστεί τις εμπορικές λύσεις σε επίπεδο enterprise χαρακτηριστικών (scalability, SLAs, πλούσια UI), αντιθέτως, καλύπτει συγκεκριμένα κενά που αυτές αδυνατούν να καλύψουν: πλήρη διαφάνεια, έλεγχο δεδομένων, ενσωματωμένο HitL feedback loop και μηδενικό λειτουργικό κόστος.

3.6 Σύνοψη Κεφαλαίου

Το παρόν κεφάλαιο τεκμηρίωσε τη δυναμική της αγοράς AI εξυπηρέτησης πελατών και ανέλυσε κριτικά τις πέντε κυρίαρχες εμπορικές πλατφόρμες. Από την ανάλυση προέκυψε ένα κοινό μοτίβο: οι υπάρχουσες λύσεις, ανεξάρτητα από το μέγεθος και την τιμή τους, λειτουργούν ως κλειστά οικοσυστήματα που περιορίζουν τον έλεγχο της επιχείρησης επί των δεδομένων της, της διαδικασίας βελτίωσης και του κόστους.

Τα τρία κενά που εντοπίστηκαν, διαφάνεια/feedback loop, κυριαρχία δεδομένων, οικονομική βιωσιμότητα, αποτελούν τον πυρήνα της Μοναδικής Αξίας (Unique Value Proposition) της παρούσας υλοποίησης. Το επόμενο κεφάλαιο παρουσιάζει τις μεθοδολογικές και αρχιτεκτονικές αποφάσεις που ελήφθησαν για να αντιμετωπιστούν αυτά τα κενά.

4. Μεθοδολογία και Αρχιτεκτονικές Αποφάσεις

Το παρόν κεφάλαιο συνδέει τη θεωρητική ανάλυση των Κεφαλαίων 2 και 3 με την πρακτική υλοποίηση που παρουσιάζεται στα Κεφάλαια 5 και 6. Παρουσιάζει και τεκμηριώνει τις κεντρικές αρχιτεκτονικές αποφάσεις που ελήφθησαν κατά το σχεδιασμό του ShopEasy ChatBot, εξηγώντας για κάθε μια τι επιλέχθηκε, τι απορρίφθηκε και γιατί. Η δομή αυτή, απόφαση, εναλλακτικές, αιτιολόγηση, αντικατοπτρίζει την επιστημονική μέθοδο: κάθε σχεδιαστική επιλογή είναι συνειδητή και ελέγξιμη, όχι αυθαίρετη.

4.1 Μεθοδολογική Προσέγγιση

Η ανάπτυξη ακολούθησε μεθοδολογία Research through Design (Zimmerman et al., 2007): το ίδιο το σύστημα λογισμικού είναι ταυτόχρονα ερευνητικό εργαλείο και αποτέλεσμα. Τα ερευνητικά ερωτήματα εξετάστηκαν μέσα από ένα λειτουργικό σύστημα σε πραγματικές συνθήκες, όχι μέσα από θεωρητικές προσομοιώσεις.

Ο κύκλος ανάπτυξης ακολούθησε τέσσερις επαναληπτικές φάσεις. Στη φάση Σχεδιασμού ορίστηκαν οι απαιτήσεις (functional requirements) βάσει των ερευνητικών ερωτημάτων και της ανάλυσης

αγοράς: τι πρέπει να κάνει το σύστημα και τι δεν χρειάζεται. Στη φάση Υλοποίησης αναπτύχθηκαν τα components με έμφαση στη σαφήνεια κώδικα και στην τεκμηρίωση. Στη φάση Αξιολόγησης δοκιμάστηκε το σύστημα με σενάρια που αντιστοιχούν απευθείας στα ερευνητικά ερωτήματα. Τέλος, στη φάση Αναστοχασμού εντοπίστηκαν οι περιορισμοί και τα σημεία βελτίωσης.

Μια κρίσιμη μεθοδολογική επιλογή ήταν η σαφής οριοθέτηση της έρευνας από την αρχή: το σύστημα σχεδιάστηκε για τοπική ανάπτυξη (local deployment), όχι παραγωγικό περιβάλλον. Αυτό επέτρεψε να επικεντρωθεί η έρευνα στα ερευνητικά ερωτήματα (αποτελεσματικότητα RAG, HitL alignment) χωρίς να αποσπά πόρους σε παράπλευρα μηχανικά ζητήματα (load balancing, high availability, security hardening).

4.2 Επιλογή Τεχνολογικού Στοιβάγματος

Η ενότητα αυτή τεκμηριώνει τις επιλογές τεχνολογιών για κάθε επίπεδο του συστήματος, παρουσιάζοντας για καθεμία τις εναλλακτικές που εξετάστηκαν και τους λόγους της τελικής απόφασης. Συγκεκριμένα, αναλύονται η επιλογή του Node.js με TypeScript για το backend, του React με TypeScript και Tailwind CSS για το frontend, και της MongoDB Atlas ως βάσης δεδομένων, η οποία αποδείχθηκε η πιο καθοριστική απόφαση καθώς επηρέασε άμεσα την αρχιτεκτονική RAG.

4.2.1 Backend: Node.js με Typescript

Το backend του συστήματος αναπτύχθηκε σε Node.js με Express και TypeScript. Η επιλογή του Node.js βασίστηκε σε τρεις παράγοντες. Πρώτον, το ασύγχρονο (async-first) μοντέλο εκτέλεσης του Node.js είναι ιδανικό για I/O-bound εφαρμογές όπως τα chatbots, όπου κάθε αίτημα απαιτεί πολλαπλές ασύγχρονες κλήσεις (βάση δεδομένων, AI API, embedding model). Δεύτερον, η χρήση JavaScript/TypeScript τόσο στο backend όσο και στο frontend (React) μειώνει την πολυπλοκότητα: ένας developer χρειάζεται να γνωρίζει μία γλώσσα για ολόκληρο το stack. Τρίτον, το npm οικοσύστημα προσφέρει ώριμες βιβλιοθήκες για κάθε component της αρχιτεκτονικής (Mongoose για MongoDB, @xenova/transformers για embeddings, @google/generative-ai για Gemini). Εναλλακτικά εξετάστηκε η Python (FastAPI/Flask), που αποτελεί τον de facto κλάδο της μηχανικής μάθησης· απορρίφθηκε ωστόσο, καθώς η ενοποίηση με το React frontend θα απαιτούσε δύο γλώσσες, ενώ το Node.js προσφέρει επαρκή υποστήριξη ML μέσω της @xenova/transformers διατηρώντας ενιαίο TypeScript stack.

Η χρήση TypeScript αντί καθαρού JavaScript αποδείχθηκε κρίσιμη: τα interfaces (π.χ. AIProvider, KnowledgeBaseDocument) τεκμηριώνουν μέσα στον ίδιο τον κώδικα τα συμβόλαια μεταξύ των components, κάτι ιδιαίτερα χρήσιμο όταν ο κώδικας πρέπει να παρουσιαστεί σε ακαδημαϊκό πλαίσιο.

4.2.2 Frontend: React με Typescript και Tailwind CSS

Το frontend αναπτύχθηκε σε React με TypeScript, bundled μέσω Vite, και styled με Tailwind CSS. Η επιλογή του React αντί εναλλακτικών (Vue, Svelte, Angular) βασίστηκε στο component model που επιτρέπει σαφή διαχωρισμό UI λογικής: κάθε component (ChatWindow, MessageBubble, FeedbackButtons, AdminPanel) αντιστοιχεί σε ένα συγκεκριμένο UI concern και μπορεί να αναπτυχθεί και δοκιμαστεί ανεξάρτητα.

Το Tailwind CSS επιλέχθηκε αντί παραδοσιακών CSS frameworks (Bootstrap, Material UI) διότι επιτρέπει γρήγορη κατασκευή custom UI χωρίς να επιβάλλει opinionated design system. Για ένα ερευνητικό πρωτότυπο όπου η λειτουργικότητα υπερέχει της αισθητικής, το utility-first approach του Tailwind είναι ιδανικό. Έναντι εναλλακτικών όπως το Vue 3 ή το Svelte, επιλέχθηκε το React 18 (με TypeScript και Tailwind CSS v4) λόγω της ευρύτερης υιοθέτησης και της πλουσιότερης βιβλιοθήκης components· το Svelte θα ήταν απλούστερο αλλά λιγότερο επεκτάσιμο.

4.2.3 Βάση Δεδομένων: MongoDB Atlas

Η επιλογή βάσης δεδομένων ήταν η πιο σημαντική αρχιτεκτονική απόφαση του συστήματος, διότι επηρέασε άμεσα την αρχιτεκτονική RAG. Εξετάστηκαν τρεις κατηγορίες λύσεων: σχεσιακές βάσεις (PostgreSQL με pgvector), εξειδικευμένες vector databases (Pinecone, Chroma, Weaviate), και MongoDB Atlas με ενσωματωμένη $vectorSearch.

Η PostgreSQL με το pgvector extension αποτελεί ισχυρή εναλλακτική: είναι ανοιχτού κώδικα, ώριμη τεχνολογία και υποστηρίζει HNSW indexing για vector search. Η ανάγκη της όμως για σχεσιακό σχήμα δεδομένων (rigid schema) έρχεται σε αντίθεση με τον δυναμικό χαρακτήρα των συνομιλιών και των embeddings, που αλλάζουν δομή καθώς εξελίσσεται το σύστημα. Και ο συνδυασμός PostgreSQL (για conversations/feedback) + pgvector (για embeddings) δημιουργεί δύο conceptual schemas, περιπλέκοντας τον κώδικα.

Οι εξειδικευμένες vector databases (Pinecone, Weaviate) προσφέρουν εξαιρετική απόδοση για μεγάλης κλίμακας vector search. Ωστόσο, εισάγουν ένα πρόσθετο σύστημα στην αρχιτεκτονική: τα conversations αποθηκεύονται στη MongoDB, τα embeddings στη Pinecone, και κάθε αλλαγή στη γνωσιακή βάση απαιτεί συγχρονισμό και των δύο. Σε production σύστημα αυτό είναι αποδεκτό, αλλά για ερευνητικό πρωτότυπο εισάγει περιττή πολυπλοκότητα. Τελικά επιλέχθηκε η MongoDB Atlas M0 με $vectorSearch έναντι των εναλλακτικών (PostgreSQL με pgvector ή Pinecone με MongoDB), καθώς προσφέρει ενιαίο αποθετήριο για συνομιλίες, feedback και embeddings, μηδενικό κόστος στο δωρεάν επίπεδο, αυτόματο fallback σε $text search για τοπική ανάπτυξη, και document-oriented δομή που ταιριάζει φυσικά στα αδόμητα δεδομένα συνομιλιών.

4.3 Επιλογή Μοντέλου Embeddings

Η επιλογή του μοντέλου παραγωγής embeddings ήταν κρίσιμη για την απόδοση του RAG pipeline. Εξετάστηκαν δύο βασικές κατευθύνσεις: cloud-based embedding APIs και τοπικά (local) μοντέλα.

4.3.1 Cloud APIs έναντι Τοπικών Μοντέλων

Οι cloud embedding APIs (OpenAI text-embedding-3-small, Google Gemini text-embedding-004) προσφέρουν υψηλής ποιότητας embeddings με εξαιρετική απόδοση σε αγγλικές εφαρμογές, εισάγουν όμως τρία προβλήματα για την παρούσα υλοποίηση. Το πρώτο είναι το κόστος: κάθε κλήση API χρεώνεται ανά token, και σε ένα σύστημα που παράγει embedding για κάθε ερώτηση χρήστη και για κάθε εγγραφή της γνωσιακής βάσης, το κόστος μπορεί να συσσωρευτεί. Ακολουθεί η εξάρτηση από το δίκτυο, καθώς ένα σύστημα που βασίζεται σε εξωτερικό API για embeddings δεν λειτουργεί offline και επηρεάζεται από κατάρρευση του παρόχου ή προβλήματα δικτύου. Υπάρχουν τέλος τα rate limits: σε φάση seeding της γνωσιακής βάσης (παραγωγή 25 embeddings ταυτόχρονα), τα όρια αιτημάτων των cloud APIs μπορεί να αποτελέσουν εμπόδιο.

Τα τοπικά μοντέλα, που εκτελούνται απευθείας στον backend server μέσω της βιβλιοθήκης @xenova/transformers, αντιμετωπίζουν και τα τρία παραπάνω προβλήματα: μηδενικό κόστος ανά κλήση, πλήρης λειτουργία offline, και απεριόριστος ρυθμός παραγωγής embeddings. Το μειονέκτημα είναι ο χρόνος φόρτωσης του μοντέλου κατά την εκκίνηση (~3-5 δευτερόλεπτα) και το αρχικό download (~250MB), το οποίο αντιμετωπίζεται με Docker volume caching. Για τους λόγους αυτούς επιλέχθηκε τοπικό μοντέλο μέσω @xenova/transformers αντί cloud API (OpenAI text-embedding-3-small ή Gemini text-embedding-004), προσφέροντας μηδενικό κόστος ανά κλήση, πλήρη offline λειτουργία και απουσία rate limits, κρίσιμα στοιχεία για το seeding των 25 εγγραφών.

4.3.2 Επιλογή Συγκεκριμένου Μοντέλου

Μεταξύ των διαθέσιμων τοπικών μοντέλων, εξετάστηκαν δύο κύριοι υποψήφιοι: το all-MiniLM-L6-v2 και το paraphrase-multilingual-MiniLM-L12-v2. Το all-MiniLM-L6-v2 είναι ελαφρύτερο (~80MB), με εξαιρετική απόδοση σε αγγλικές εφαρμογές και ευρεία χρήση στην κοινότητα. Υστερεί όμως στην υποστήριξη της ελληνικής γλώσσας, η οποία είναι περιορισμένη.

Το paraphrase-multilingual-MiniLM-L12-v2 υποστηρίζει 50+ γλώσσες συμπεριλαμβανομένων των ελληνικών, και, κρίσιμα, παρέχει cross-lingual semantic matching: μια ερώτηση στα αγγλικά μπορεί να αντιστοιχιστεί με έγγραφο στα ελληνικά (Reimers & Gurevych, 2020). Αυτό σημαίνει ότι η γνωσιακή βάση μπορεί να παραμείνει αποκλειστικά στα ελληνικά, ενώ το σύστημα χειρίζεται ερωτήσεις και στις δύο γλώσσες. Το μοντέλο είναι ελαφρώς μεγαλύτερο (~250MB) και παράγει εξίσου διανύσματα 384 διαστάσεων. Επιλέχθηκε επομένως το paraphrase-multilingual-MiniLM-L12-v2 (384 διαστάσεις) έναντι του ελαφρύτερου all-

MiniLM-L6-v2, χάρη στην υποστήριξη της ελληνικής και στο cross-lingual matching, που επιτρέπει σε αγγλικές ερωτήσεις να αντιστοιχίζονται σωστά με ελληνικές εγγραφές χωρίς ρητή μετάφραση.

4.4 Αφαίρεση Παρόχου AI (Provider Abstraction)

Μια από τις πιο σημαντικές αρχιτεκτονικές αποφάσεις ήταν ο σχεδιασμός του συστήματος ώστε να είναι αγνωστικό ως προς τον πάροχο AI (ανεξάρτητο από πάροχο). Αντί να συνδεθεί άμεσα με έναν πάροχο (π.χ. μόνο Gemini), ορίστηκε ένα κοινό interface (AIProvider) που όλοι οι πάροχοι πρέπει να υλοποιούν. Ένα Factory Pattern αποφασίζει ποιον πάροχο να φορτώσει βάσει μιας περιβαλλοντικής μεταβλητής.

4.4.1 Το AIProvider Interface

Το interface ορίζει μία μόνο μέθοδο: GenerateResponse(messages, systemPrompt). Αυτή η απλότητα είναι σκόπιμη: αποκρύπτει εντελώς τις διαφορές μεταξύ παρόχων (διαφορετικά SDK, διαφορετικά formats μηνυμάτων, διαφορετικά rate limits) πίσω από ένα κοινό συμβόλαιο. Ο κώδικας του chatService που καλεί το AI δεν γνωρίζει, και δεν χρειάζεται να γνωρίζει, αν μιλά με Gemini, OpenAI ή OpenRouter.

Αυτή η αρχιτεκτονική, γνωστή ως αρχή Dependency Inversion (ένα από τα SOLID principles της αντικειμενοστρεφούς ανάπτυξης), εξασφαλίζει ότι η αλλαγή παρόχου δεν απαιτεί τροποποίηση κεντρικής λογικής, μόνο μια γραμμή στο .env αρχείο. Αυτό έχει σημαντική ερευνητική αξία: επιτρέπει τη σύγκριση διαφορετικών μοντέλων στις ίδιες συνθήκες χωρίς αλλαγή κώδικα.

4.4.2 Υποστηριζόμενοι Πάροχοι

Υλοποιήθηκαν τρεις πάροχοι AI: OpenRouter, Google Gemini και OpenAI. Ο OpenRouter αποτελεί τον προεπιλεγμένο πάροχο για ανάπτυξη, καθώς προσφέρει ενιαία πρόσβαση σε δεκάδες μοντέλα μέσω ενός API key και δωρεάν tier. Ο Google Gemini (συγκεκριμένα το Gemini 2.5 Flash) συνιστάται για το demo, λόγω εξαιρετικής ισορροπίας ποιότητας και δωρεάν ορίων χρήσης. Ο OpenAI (GPT-4o-mini ή GPT-4o) παρέχει την υψηλότερη ποιότητα αλλά με κόστος ανά token.

Πίνακας 4.1: Σύγκριση υποστηριζόμενων παρόχων AI

Πάροχος	Μοντέλο (default)	Κόστος	Πλεονέκτημα	Μειονέκτημα
OpenRouter	Llama 3.3 70B	Δωρεάν tier	Πρόσβαση σε πολλά μοντέλα	Μεταβλητή απόδοση
Google Gemini	Gemini 2.5 Flash	Δωρεάν tier (γενναιόδωρο)	Γρήγορο, υψηλή ποιότητα	Απαιτεί Google account
OpenAI	GPT-4o-mini	Χρέωση ανά token	Υψηλότερη ποιότητα	Κόστος σε παραγωγή

Πηγή: Τεχνική τεκμηρίωση παρόχων (2025).

4.5 Σχεδιασμός του RAG Pipeline

Πέρα από την επιλογή των επιμέρους τεχνολογιών, η αποτελεσματικότητα του RAG εξαρτάται από τις παραμέτρους σχεδιασμού του pipeline. Η ενότητα αυτή παρουσιάζει τις βασικές αυτές παραμέτρους, δηλαδή τον αριθμό των εγγράφων που ανακτώνται, το κατώφλι ομοιότητας και τον αριθμό υποψηφίων του αλγορίθμου αναζήτησης, και αιτιολογεί τις τιμές που επιλέχθηκαν. Στη συνέχεια περιγράφεται η στρατηγική fallback που επιτρέπει στο σύστημα να λειτουργεί τόσο σε περιβάλλον MongoDB Atlas όσο και σε τοπική εγκατάσταση.

4.5.1 Βασικές Παράμετροι Σχεδιασμού

Το RAG pipeline του συστήματος έχει τρεις βασικές παραμέτρους που επηρεάζουν άμεσα την απόδοσή του: τον αριθμό εγγράφων που ανακτώνται (Top-N), το κατώφλι ομοιότητας (Similarity Threshold), και τον αριθμό υποψηφίων που εξετάζει ο HNSW αλγόριθμος (numCandidates).

Για τον αριθμό εγγράφων (Top-N), επιλέχθηκε τιμή N=3. Η επιλογή αυτή αντισταθμίζει δύο αντικρουόμενες ανάγκες: από τη μία, η χρήση περισσότερων εγγράφων (N=5 ή N=10) αυξάνει την πιθανότητα συμπερίληψης σχετικής πληροφορίας, αλλά αυξάνει επίσης το μέγεθος του context, το κόστος tokens, και τον κίνδυνο «αραίωσης» της σχετικής πληροφορίας μεταξύ άσχετων εγγράφων. Από την άλλη, N=1 ή N=2 ενδέχεται να χάσει σχετικές πληροφορίες που βρίσκονται κατανεμημένες σε πολλά έγγραφα. Το N=3 αποτελεί την πλέον συνηθισμένη τιμή στη βιβλιογραφία RAG για βάσεις γνώσης μέτριου μεγέθους (Gao et al., 2024).

Για το κατώφλι ομοιότητας, επιλέχθηκε τιμή 0,5 σε κλίμακα 0–1. Τιμή κατωφλίου κάτω από 0,4 οδηγεί σε συμπερίληψη σχεδόν άσχετων εγγράφων που μπορεί να παραπλανήσουν το μοντέλο. Τιμή άνω από 0,65 είναι πολύ περιοριστική για μια γλώσσα όπως τα ελληνικά, όπου οι παραφράσεις μπορεί να έχουν χαμηλότερη ομοιότητα λόγω της πλούσιας μορφολογίας. Η τιμή 0,5 παρέχει ισορροπία μεταξύ ακρίβειας (precision) και ανάκλησης (recall).

4.5.2 Fallback Στρατηγική: Atlas έναντι Local MongoDB

Ένα από τα πιο πρακτικά ζητήματα σχεδιασμού ήταν η διαχείριση της ασυμβατότητας: ο τελεστής $vectorSearch της MongoDB λειτουργεί αποκλειστικά στο MongoDB Atlas, όχι σε τοπικές εγκαταστάσεις. Αυτό σημαίνει ότι ένας developer που θέλει να τρέξει το σύστημα τοπικά με Docker χωρίς Atlas connection δεν θα έχει RAG.

Η λύση που υιοθετήθηκε είναι η αυτόματη ανίχνευση περιβάλλοντος και fallback: το ragService εξετάζει αν η συμβολοσειρά σύνδεσης MONGODB_URI αρχίζει με mongodb+srv:// (ένδειξη Atlas σύνδεσης). Εάν ναι, χρησιμοποιεί $vectorSearch. Εάν όχι, ενεργοποιεί $text search, μια παραδοσιακή αναζήτηση με λέξεις-κλειδιά που βασίζεται σε text index της MongoDB. Αυτή η προσέγγιση διασφαλίζει ότι η ίδια codebase λειτουργεί σε και τα δύο περιβάλλοντα, με αυτόματη

επιλογή της βέλτιστης μεθόδου. Έτσι αποφεύγεται τόσο η απαίτηση μόνιμης σύνδεσης με το Atlas όσο και ο περιορισμός σε αποκλειστικά τοπική λειτουργία: η ίδια codebase προσαρμόζεται αυτόματα μεταξύ περιβάλλοντος ανάπτυξης και demo.

4.6 Σχεδιασμός του Human-in-the-Loop Μηχανισμού

Ο μηχανισμός Human-in-the-Loop αποτελεί το δεύτερο κεντρικό τεχνικό στοιχείο της εργασίας, μέσω του οποίου το σύστημα βελτιώνεται από ανθρώπινες διορθώσεις χωρίς επανεκπαίδευση. Η ενότητα αυτή παρουσιάζει πρώτα τη φιλοσοφία σχεδιασμού του μηχανισμού, στη συνέχεια αναλύει τα τέσσερα στάδια του κύκλου ανατροφοδότησης από τη συλλογή της αξιολόγησης έως την έγχυση της διόρθωσης, και τέλος εξετάζει τους τεχνικούς περιορισμούς που αναγνωρίστηκαν κατά τον σχεδιασμό και τον τρόπο με τον οποίο αντιμετωπίστηκαν.

4.6.1 Φιλοσοφία Σχεδιασμού

Ο HitL μηχανισμός σχεδιάστηκε με τρεις απαιτήσεις: ο διαχειριστής να βλέπει ποιες ερωτήσεις αποτυγχάνουν (ορατότητα), να αποφασίζει ποιες διορθώσεις εφαρμόζονται (έλεγχος), και οι εγκεκριμένες διορθώσεις να επηρεάζουν το σύστημα στο επόμενο request χωρίς restart (άμεση εφαρμογή).

Αυτή η φιλοσοφία αντιτίθεται άμεσα στον τρόπο λειτουργίας των εμπορικών πλατφορμών (βλ. Κεφάλαιο 3), όπου ο κύκλος βελτίωσης είναι είτε κλειστός (black box) είτε αργός (απαιτεί εβδομάδες για re-training). Η παρούσα υλοποίηση επιδεικνύει ότι ο κύκλος μπορεί να είναι δευτερολέπτων: χρήστης δίνει αρνητική αξιολόγηση → διαχειριστής εγκρίνει → επόμενος χρήστης λαμβάνει βελτιωμένη απάντηση.

4.6.2 Ο Κύκλος Ανατροφοδότησης: Τεχνική Υλοποίηση

Ο κύκλος ανατροφοδότησης αποτελείται από τέσσερα διακριτά στάδια που αντιστοιχούν σε διαφορετικά components του συστήματος. Στο πρώτο στάδιο (Συλλογή), ο χρήστης κάνει κλικ στο κουμπί 👎 και προαιρετικά πληκτρολογεί τη σωστή απάντηση. Η ερώτηση του χρήστη, η λανθασμένη απάντηση του bot, και η πρότειλη διόρθωση αποθηκεύονται στη συλλογή feedbacks της MongoDB με status: "pending".

Στο δεύτερο στάδιο (Αναθεώρηση), ο διαχειριστής επισκέπτεται το admin panel και βλέπει όλες τις εκκρεμείς αξιολογήσεις. Για κάθε εγγραφή, βλέπει τη ερώτηση του χρήστη, την απάντηση του bot (επισημασμένη με κόκκινο), και το πεδίο διόρθωσης. Μπορεί να επεξεργαστεί τη διόρθωση πριν την εγκρίνει, ή να την απορρίψει εάν η αρχική απάντηση ήταν σωστή.

Στο τρίτο στάδιο (Έγκριση), το click «Approve» αλλάζει το status της εγγραφής σε "approved". Από αυτή τη στιγμή, η εγγραφή γίνεται «χρυσός κανόνας» (golden rule), ένα few-shot παράδειγμα ζεύγους λάθους/σωστής απάντησης.

Στο τέταρτο στάδιο (Έγχυση), σε κάθε νέο αίτημα chat, το feedbackEngine ανακτά τους χρυσούς κανόνες από τη MongoDB (έως 5) και τους εισάγει στο system prompt ως few-shot examples. Το LLM «βλέπει» τα παραδείγματα και, χάρη στις δυνατότητες in-context learning, αποφεύγει να κάνει τα ίδια λάθη.

4.6.3 Τεχνικοί Περιορισμοί και πως Αντιμετωπίστηκαν

Ο μηχανισμός HitL παρουσιάζει δύο δομικούς περιορισμούς που αναγνωρίστηκαν κατά το σχεδιασμό και αντιμετωπίστηκαν ρητά. Ο πρώτος είναι ο περιορισμός context window: η εισαγωγή πολλών κανόνων στο system prompt αυξάνει τον αριθμό tokens ανά αίτημα, αυξάνοντας κόστος και λανθάνοντα χρόνο. Αντιμετωπίστηκε με όριο: έως 5 χρυσοί κανόνες και έως 5 αρνητικά παραδείγματα εισάγονται ταυτόχρονα.

Ο δεύτερος περιορισμός είναι η «αραίωση» (dilution): εάν ένας κανόνας αφορά ένα πολύ ειδικό θέμα, η παρουσία του στο system prompt κάθε αιτήματος (ακόμα και άσχετων) είναι ελαφρώς ανεπίστρεπτη σπατάλη context. Μια πιο εξελιγμένη υλοποίηση θα εισήγαγε μόνο τους κανόνες που είναι σχετικοί με την τρέχουσα ερώτηση (rule retrieval). Αυτό προτείνεται ως μελλοντική επέκταση στο Κεφάλαιο 8.

4.7 Υποδομή: Docker και Περιβαλλοντικές Μεταβλητές

Η επιλογή του Docker ως υποδομή ανάπτυξης δεν ήταν αυτονόητη για ένα ερευνητικό πρωτότυπο, αποδείχθηκε ωστόσο κρίσιμη για δύο λόγους. Ο πρώτος αφορά την αναπαραγωγιμότητα: ένα αρχείο Docker Compose εξασφαλίζει ότι το σύστημα εκτελείται ταυτόσημα σε οποιοδήποτε μηχάνημα, κάτι κρίσιμο για την ακαδημαϊκή αξιολόγηση. Ο δεύτερος αφορά τη διαχείριση του μοντέλου embeddings: το τοπικό μοντέλο (~250MB) κατεβαίνει μόνο μία φορά και αποθηκεύεται σε Docker volume, αποφεύγοντας το επανακατέβασμα σε κάθε επανεκκίνηση.

Η αρχιτεκτονική μεταβλητών περιβάλλοντος (environment variables) ακολουθεί την αρχή των Twelve-Factor Apps (Wiggins, 2017): όλες οι ρυθμίσεις που διαφέρουν μεταξύ περιβαλλόντων (development, demo, production) αποθηκεύονται σε .env αρχείο και ποτέ δεν hardcoded στον κώδικα. Αυτό περιλαμβάνει τα API keys, τη σύνδεση βάσης δεδομένων, τον πάροχο AI, και τον κωδικό admin.

Πίνακας 4.2: Κεντρικές περιβαλλοντικές μεταβλητές και ο ρόλος τους

Μεταβλητή	Παράδειγμα τιμής	Ρόλος στη Αρχιτεκτονική
MONGODB_URI	mongodb+srv://...	Καθορίζει Atlas vs local — ενεργοποιεί $vectorSearch ή $text fallback
AI_PROVIDER	gemini	Καθορίζει ποιον πάροχο φορτώνει το Factory Pattern
GEMINI_API_KEY	AIza...	Κλειδί Gemini API — χρησιμοποιείται μόνο εάν AI_PROVIDER=gemini
GEMINI_MODEL	gemini-2.5-flash	Το συγκεκριμένο μοντέλο εντός του παρόχου
ADMIN_PASSWORD	admin	Κωδικός admin panel — απλός για demo, θα ήταν bcrypt σε παραγωγή
PORT	3000	Η θύρα στην οποία ακούει το Express backend

Πηγή: Ιδία επεξεργασία.

4.8 Σύνοψη Αρχιτεκτονικών Αποφάσεων

Ο Πίνακας 4.3 συνοψίζει το σύνολο των αρχιτεκτονικών αποφάσεων που παρουσιάστηκαν στο παρόν κεφάλαιο, αποτυπώνοντας τη λογική αλληλεξάρτησή τους.

Πίνακας 4.3: Σύνοψη αρχιτεκτονικών αποφάσεων

Τομέας	Επιλογή	Κύρια Αιτιολόγηση
Backend Runtime	Node.js 18 + TypeScript	Ασύγχρονο I/O, ενιαίο JS stack, ισχυρό npm οικοσύστημα
Frontend	React 18 + TypeScript + Tailwind	Component model, ευρεία υιοθέτηση, γρήγορη ανάπτυξη UI
Βάση Δεδομένων	MongoDB Atlas M0	Ενιαία αποθήκευση docs + vectors, δωρεάν tier, document model

Τομέας	Επιλογή	Κύρια Αιτιολόγηση
Embeddings	paraphrase-multilingual-MiniLM-L12-v2	Τοπική εκτέλεση, ελληνική γλώσσα, cross-lingual matching
Vector Search	MongoDB $vectorSearch + $text fallback	Δεν χρειάζεται ξεχωριστό vector DB, αυτόματο fallback
AI Provider	Factory Pattern (Gemini / OpenRouter / OpenAI)	Αναλλαξιμότητα παρόχου μέσω ENV, χωρίς αλλαγή κώδικα
HitL Alignment	Few-shot injection σε system prompt	Inference-time alignment, άμεση εφαρμογή, χωρίς επανεκπαίδευση
Υποδομή	Docker Compose + .env αρχεία	Αναπαραγωγιμότητα, caching μοντέλου, Twelve-Factor compliance

Πηγή: Ιδία επεξεργασία.

Οι αποφάσεις αυτές δεν λήφθηκαν μεμονωμένα· αποτελούν μέρος μιας συνεκτικής αρχιτεκτονικής λογικής: ελαχιστοποίηση εξωτερικών εξαρτήσεων, ευελιξία στην αλλαγή παρόχου, και εστίαση στα ερευνητικά ερωτήματα. Τα επόμενα κεφάλαια δείχνουν πώς υλοποιήθηκαν αυτές οι επιλογές.

5. Σχεδιασμός Συστήματος

Το κεφάλαιο αυτό παρουσιάζει την αρχιτεκτονική σε τεχνικό επίπεδο: σχήματα δεδομένων, API endpoints, ροές εκτέλεσης και οργάνωση κώδικα. Οι αποφάσεις τεκμηριώθηκαν στο Κεφάλαιο 4· εδώ εξετάζεται η δομή που τις υλοποιεί.

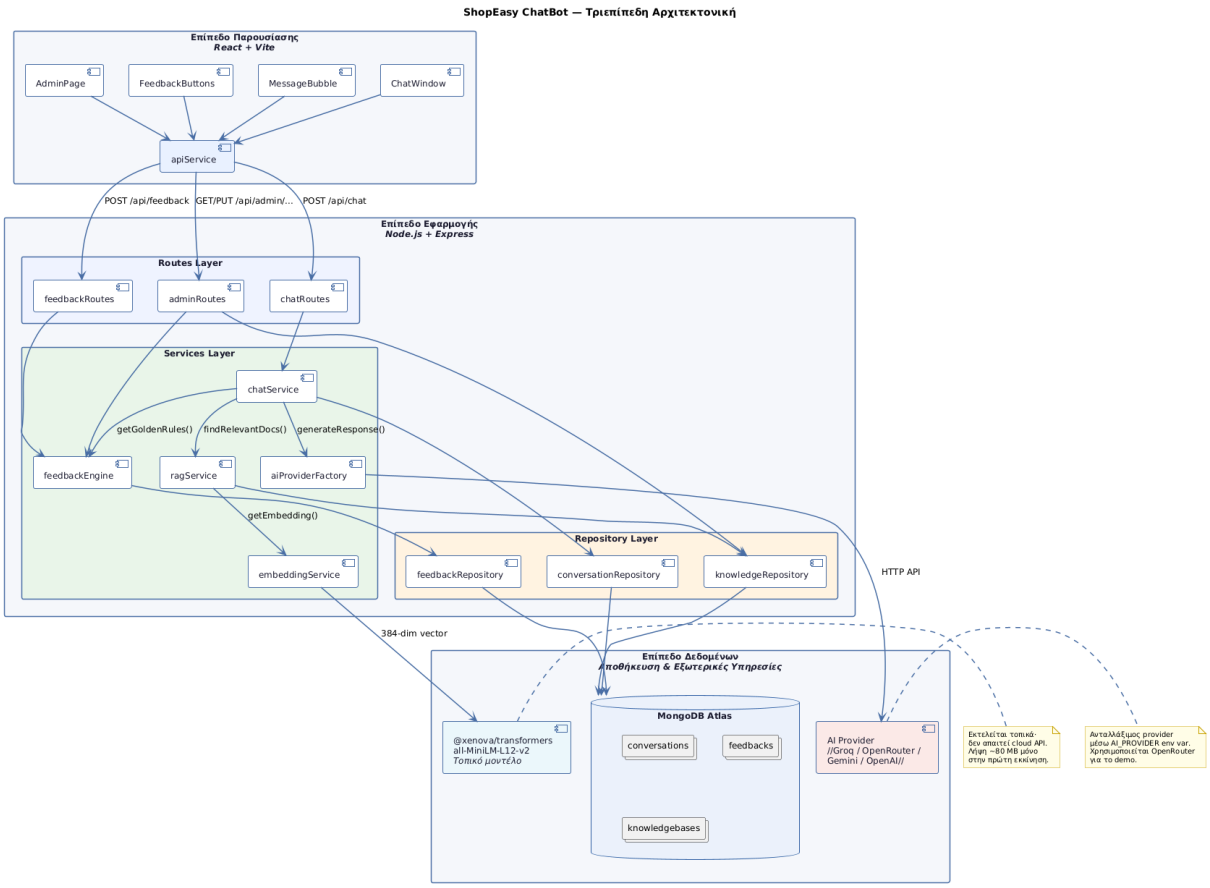
5.1 Επισκόπηση Τριεπίπεδης Αρχιτεκτονικής

Το σύστημα ακολουθεί κλασική τριεπίπεδη αρχιτεκτονική (3-tier architecture) με σαφή διαχωρισμό ευθυνών μεταξύ των τριών επιπέδων: παρουσίαση, επιχειρησιακή λογική και δεδομένα. Ο διαχωρισμός αυτός προσφέρει συγκεκριμένα πλεονεκτήματα στην πράξη: κάθε επίπεδο μπορεί να τροποποιηθεί ανεξάρτητα, η δοκιμή κάθε επιπέδου γίνεται χωρίς εξάρτηση από τα άλλα, και η αντικατάσταση ενός επιπέδου (π.χ. αλλαγή παρόχου AI) δεν επηρεάζει τα υπόλοιπα.

Πίνακας 5.1: Τα τρία επίπεδα του συστήματος

Επίπεδο	Τεχνολογία	Ευθύνη	Port
Παρουσίαση (Frontend)	React 18 + TypeScript + Vite	Chat UI, Admin Panel, Feedback buttons	5173
Επιχειρησιακή Λογική (Backend)	Node.js + Express + TypeScript	RAG pipeline, AI orchestration, Feedback loop	3000
Δεδομένα (Database)	MongoDB Atlas / Docker	Conversations, Feedbacks, Knowledge Base + Vectors	27017

Πηγή: Ιδία επεξεργασία.



Διάγραμμα 5.1: Τριεπίπεδη Αρχιτεκτονική

Η επικοινωνία μεταξύ frontend και backend γίνεται αποκλειστικά μέσω REST API κλήσεων (HTTP/JSON). Δεν χρησιμοποιείται WebSocket, επιλογή που απλοποιεί την αρχιτεκτονική για ερευνητικό πρωτότυπο, με το κόστος ότι ο χρήστης βλέπει μία ενδιάμεση κινούμενη ένδειξη αναμονής αντί για streaming. Το backend επικοινωνεί με δύο εξωτερικές υπηρεσίες: τον πάροχο AI (μέσω HTTP) και τη MongoDB (μέσω TCP/Mongoose).

5.2 Δομή Κώδικα

Η οργάνωση του κώδικα ακολουθεί αρχιτεκτονικό μοτίβο που διαχωρίζει σαφώς την επιχειρησιακή λογική (services) από την αλληλεπίδραση με τα δεδομένα (repositories), τα ορισμένα σχήματα (models), τα σημεία εισόδου HTTP (routes) και τα βοηθητικά scripts. Αυτός ο διαχωρισμός συμβάλλει στη δυνατότητα επέκτασης: η αλλαγή βάσης δεδομένων θα επηρεάσει μόνο το repository layer, ενώ η αλλαγή AI provider μόνο το service layer.

Πίνακας 5.2: Δομή αρχείων backend

Φάκελος / Αρχείο	Ρόλος
config/aiProviderFactory.ts	Factory Pattern — επιλέγει πάροχο AI βάσει ENV

Φάκελος / Αρχείο	Ρόλος
config/database.ts	Σύνδεση MongoDB μέσω Mongoose
config/env.ts	Validation μεταβλητών περιβάλλοντος κατά την εκκίνηση
models/Conversation.ts	Mongoose schema για τις συνομιλίες
models/Feedback.ts	Mongoose schema για τις αξιολογήσεις χρηστών
models/KnowledgeBase.ts	Mongoose schema για τη γνωσιακή βάση (+ embeddings)
repositories/conversationRepository.ts	CRUD για συνομιλίες (GetSessionHistory, SaveMessage)
repositories/feedbackRepository.ts	CRUD για feedbacks (SaveFeedback, UpdateStatus)
repositories/knowledgeRepository.ts	CRUD για γνωσιακή βάση
repositories/adminRepository.ts	Aggregation queries για admin panel (stats, feedback με context)
routes/chatRoutes.ts	POST /api/chat — είσοδος μηνυμάτων
routes/feedbackRoutes.ts	POST /api/feedback — υποβολή αξιολογήσεων
routes/adminRoutes.ts	GET/PUT /api/admin/feedback, CRUD /api/admin/knowledge
services/chatService.ts	Ενορχήστρωση: RAG → prompt → AI → save → return
services/ragService.ts	Vector/text search στη γνωσιακή βάση
services/embeddingService.ts	Τοπική παραγωγή embeddings (@xenova/transformers)
services/feedbackEngine.ts	Μετατροπή feedback σε few-shot examples για system prompt
services/AIProvider.ts	Interface: GenerateResponse(messages, systemPrompt)
services/GeminiProvider.ts	Υλοποίηση Gemini
services/OpenAIProvider.ts	Υλοποίηση OpenAI
services/OpenRouterProvider.ts	Υλοποίηση OpenRouter
scripts/seedKnowledge.ts	One-time script: φορτώνει 25 FAQ με auto-generated embeddings

Πηγή: Ιδία επεξεργασία.

5.3 Σχήματα Δεδομένων MongoDB

Η MongoDB χρησιμοποιεί document model (αντί για σχεσιακό), που σημαίνει ότι κάθε "σχήμα" αντιστοιχεί σε μια συλλογή (collection) εγγράφων με ευέλικτη δομή. Το σύστημα χρησιμοποιεί τρεις κύριες συλλογές: conversations, feedbacks, και knowledgebases. Το

Mongoose (ODM για Node.js) επιβάλλει τη δομή κατά την εγγραφή, παρέχοντας στατική τυποποίηση μέσω TypeScript interfaces.

5.3.1 Συλλογή conversations

Κάθε έγγραφο αντιστοιχεί σε ένα session χρήστη και περιέχει ολόκληρο το ιστορικό της συνομιλίας ως embedded array μηνυμάτων. Η επιλογή ενσωματωμένου array αντί για ξεχωριστή συλλογή messages αποφεύγει joins και επιτρέπει ανάκτηση ολόκληρης της συνομιλίας με ένα read operation.

Πίνακας 5.3: Σχήμα συλλογής conversations

Πεδίο	Τύπος	Υποχρεωτικό	Περιγραφή
_id	ObjectId	Αυτόματο	MongoDB εσωτερικό ID
sessionId	String	Ναι	UUID v4 — μοναδικό ανά συνομιλία, indexed
messages[].messageId	String	Ναι	UUID v4 — αναφέρεται στο feedback
messages[].role	String	Ναι	"user" | "assistant" | "system"
messages[].content	String	Ναι	Κείμενο μηνύματος
messages[].timestamp	Date	Ναι	Χρόνος δημιουργίας
createdAt	Date	Αυτόματο	Mongoose timestamps
updatedAt	Date	Αυτόματο	Ενημερώνεται σε κάθε νέο μήνυμα

Πηγή: Ιδία επεξεργασία.

5.3.2 Συλλογή feedbacks

Κάθε έγγραφο αντιστοιχεί σε μια αξιολόγηση χρήστη (👍 ή 👎) για μια συγκεκριμένη απάντηση του bot. Κρίσιμο σχεδιαστικό στοιχείο: το userQuestion και το botAnswer αποθηκεύονται απευθείας στο feedback document κατά την υποβολή (de-normalization). Αυτό αποφεύγει την ανάγκη για join με τη συλλογή conversations κατά την εμφάνιση στο admin panel, βελτιώνοντας τον χρόνο ανάκτησης.

Πίνακας 5.4: Σχήμα συλλογής feedbacks

Πεδίο	Τύπος	Default	Περιγραφή
_id	ObjectId	—	MongoDB εσωτερικό ID
messageId	String	required	ID απάντησης bot — αντιστοιχεί σε messages[].messageId
sessionId	String	required	ID συνομιλίας
rating	Number	required	+1 (👍) ή -1 (👎)
userQuestion	String	null	Ερώτηση που προηγήθηκε — αποθ. τη στιγμή υποβολής
botAnswer	String	null	Απάντηση bot που αξιολογήθηκε
correction	String	null	Σωστή απάντηση από χρήστη ή admin
status	String	"pending"	"pending" | "approved" | "rejected"
category	String	null	Κατηγορία θέματος (προαιρετικό)
createdAt	Date	Αυτόματο	Mongoose timestamps

Πηγή: Ιδία επεξεργασία.

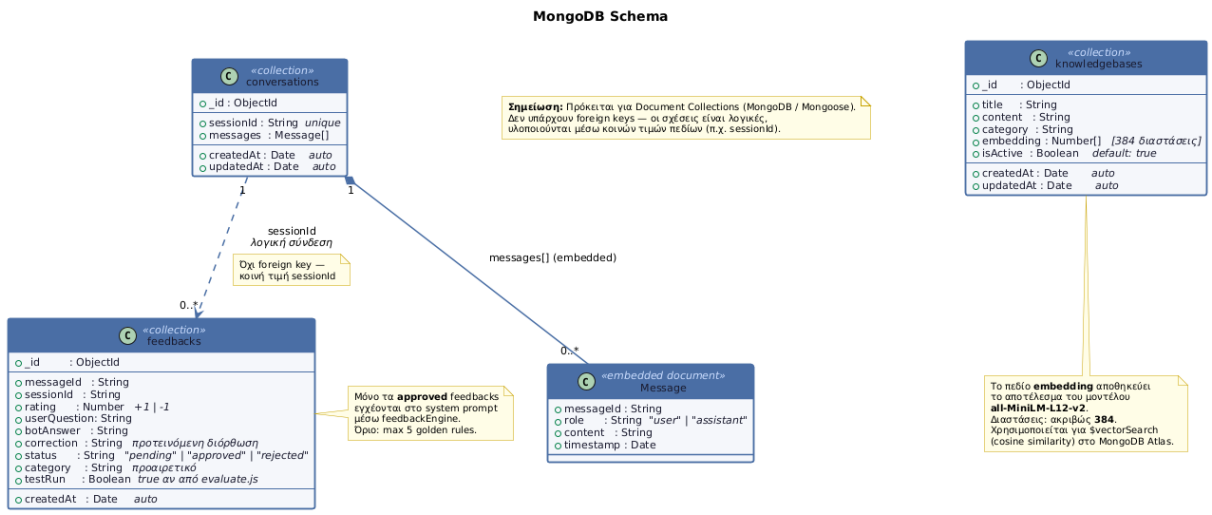
5.3.3 Συλλογή knowledgebases

Κάθε έγγραφο αντιστοιχεί σε μια εγγραφή της γνωσιακής βάσης (FAQ, πολιτική, οδηγίες) και περιέχει τόσο το αναγνώσιμο κείμενο (content) όσο και τον αριθμητικό του «αποτύπωμα» (embedding). Η συνύπαρξη κειμένου και embedding στο ίδιο document είναι η κεντρική αρχιτεκτονική επιλογή που επιτρέπει την ενοποιημένη αποθήκευση χωρίς ξεχωριστή vector database. Το πεδίο isActive υλοποιεί soft delete, τα "διαγραμμένα" έγγραφα παραμένουν στη βάση αλλά εξαιρούνται από αναζητήσεις.

Πίνακας 5.5: Σχήμα συλλογής knowledgebases

Πεδίο	Τύπος	Υποχρεωτικό	Περιγραφή
_id	ObjectId	Αυτόματο	MongoDB εσωτερικό ID
title	String	Ναι	Τίτλος εγγραφής (π.χ. "Πολιτική Επιστροφών")
content	String	Ναι	Πλήρες κείμενο FAQ — αυτό εμφανίζεται στο system prompt
category	String	Ναι	"returns" | "shipping" | "payments" | "products" | "account"
embedding	Number[]	Ναι	384 αριθμοί float — διάνυσμα σημασιολογικής αναπαράστασης
isActive	Boolean	Ναι	false = soft deleted, εξαιρείται από RAG
createdAt	Date	Αυτόματο	Mongoose timestamps
updatedAt	Date	Αυτόματο	Ενημερώνεται αν αλλάξει το content ή η κατηγορία

Πηγή: Ιδία επεξεργασία.



Διάγραμμα 5.2: MongoDB Schema

5.4 API Contracts

Το σύστημα εκθέτει τρία groups endpoints που αντιστοιχούν σε τρεις λειτουργικές ενότητες: το chat (αλληλεπίδραση χρηστών), το feedback (αξιολόγηση απαντήσεων) και το admin (διαχείριση). Όλα τα endpoints χρησιμοποιούν JSON ως format επικοινωνίας και επιστρέφουν κατάλληλους HTTP status codes (200 για επιτυχία, 400 για λάθος εισόδου, 500 για εσωτερικό σφάλμα).

Πίνακας 5.6: REST API endpoints

Method	Path	Σκοπός	Request Body	Response
POST	/api/chat	Υποβολή μηνύματος χρήστη	{ message, sessionId? }	{ sessionId, reply, messageId }
POST	/api/feedback	Υποβολή αξιολόγησης	{ messageId, sessionId, vote, userQuestion?, botAnswer?, correction? }	{ success: true }
GET	/api/admin/feedback	Λίστα αρνητικών αξιολογήσεων	—	FeedbackDocument[]
GET	/api/admin/feedback/stats	Στατιστικά αξιολογήσεων	—	{ total, positivePercent, negativePercent, pending, approved, rejected }
PUT	/api/admin/feedback/:messageId	Έγκριση / Απόρριψη	{ status, correction? }	{ success: true }
GET	/api/admin/knowledge	Λίστα εγγραφών γνωσιακής βάσης	—	KnowledgeBaseDocument[] (χωρίς embeddings)
POST	/api/admin/knowledge	Νέα εγγραφή (auto-embed)	{ title, content, category }	KnowledgeBaseDocument
PUT	/api/admin/knowledge/:id	Επεξεργασία (re-embed αν αλλάξει content)	{ title?, content?, category? }	KnowledgeBaseDocument
DELETE	/api/admin/knowledge/:id	Soft delete (isActive: false)	—	{ success: true }

Πηγή: Ιδία επεξεργασία.

Μια σημαντική λεπτομέρεια στο API design αφορά το /api/admin/feedback/stats endpoint: πρέπει να ορίζεται πριν από το /api/admin/feedback/:messageId στον Express router,

διαφορετικά το Express ερμηνεύει το "stats" ως τιμή για το parameter messageId. Αυτή είναι μια κλασική παγίδα στο REST routing που αντιμετωπίζεται με συνειδητή σειρά ορισμού των routes.

5.5 Ροές Δεδομένων

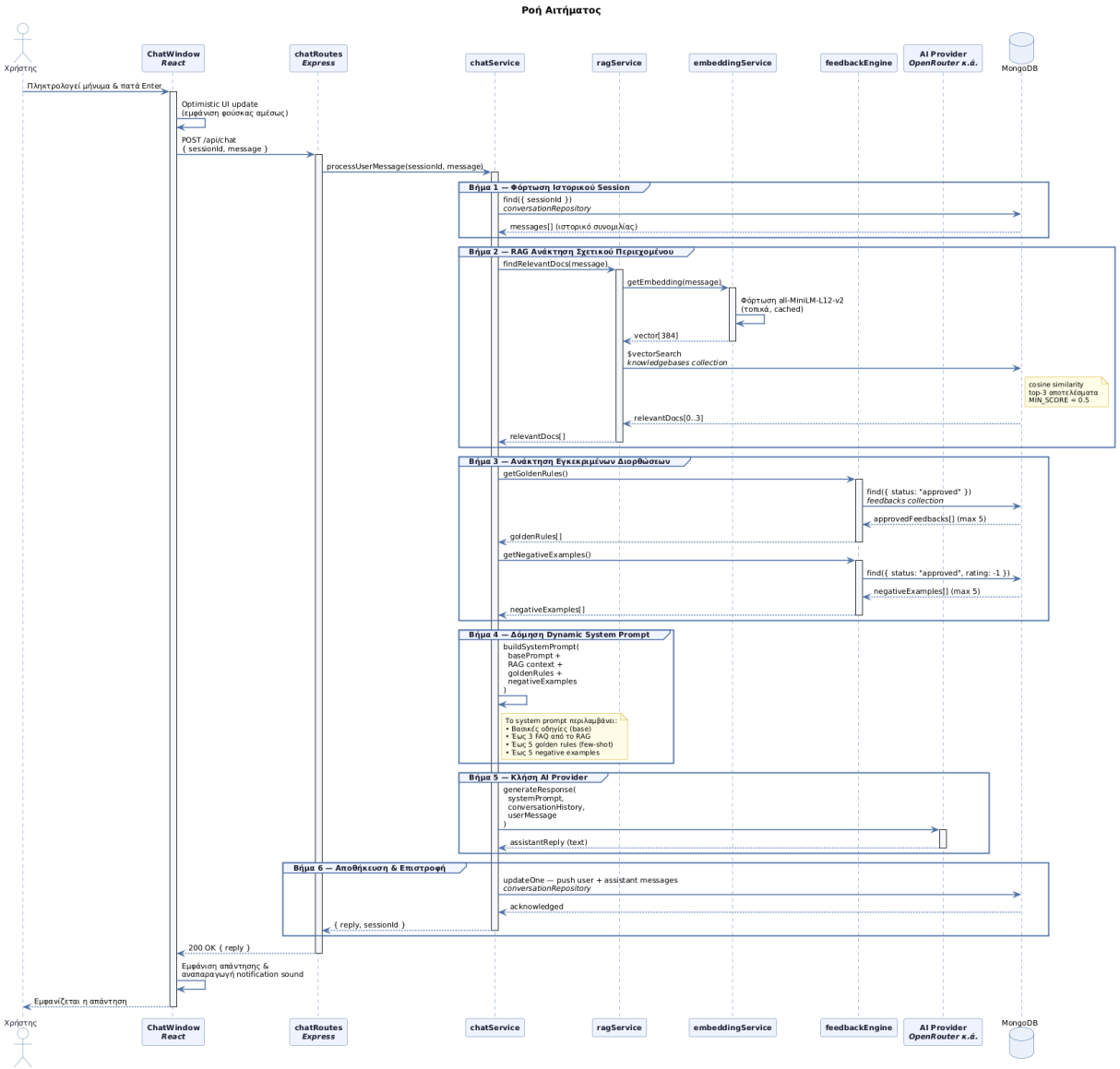
5.5.1 Ροή Chat Request (End-to-End)

Η κεντρική ροή του συστήματος, η επεξεργασία ενός μηνύματος χρήστη, περιλαμβάνει έξι διακριτά βήματα που εκτελούνται σειριακά. Η σειριακή εκτέλεση (αντί παράλληλης) διασφαλίζει ότι κάθε βήμα λαμβάνει τα αποτελέσματα του προηγούμενου: το RAG context πρέπει να είναι έτοιμο πριν χτιστεί το prompt, και το prompt πριν κληθεί το AI.

Πίνακας 5.7: Βήματα επεξεργασίας chat request

Βήμα	Component	Είσοδος	Έξοδος	Αποτυχία
1. Φόρτωση ιστορικού	conversationRepository	sessionId	Message[] ή []	Επιστρέφει []
2. RAG αναζήτηση	ragService	userMessage (text)	RAGResult[] (έως 3 έγγραφα)	Επιστρέφει []
3. Κατασκευή prompt	chatService (BuildSystemPrompt)	RAG context + feedbacks	system prompt (string)	Δεν αποτυγχάνει
4. Κλήση AI	AIProvider.GenerateResponse	messages[] + systemPrompt	reply (string)	throw Error
5. Αποθήκευση	conversationRepository	sessionId + messages	—	throw Error
6. Επιστροφή	chatRoutes	reply + messageId	HTTP 200 JSON	—

Πηγή: Ιδία επεξεργασία.



Διάγραμμα 5.3: Ροή Αιτήματος

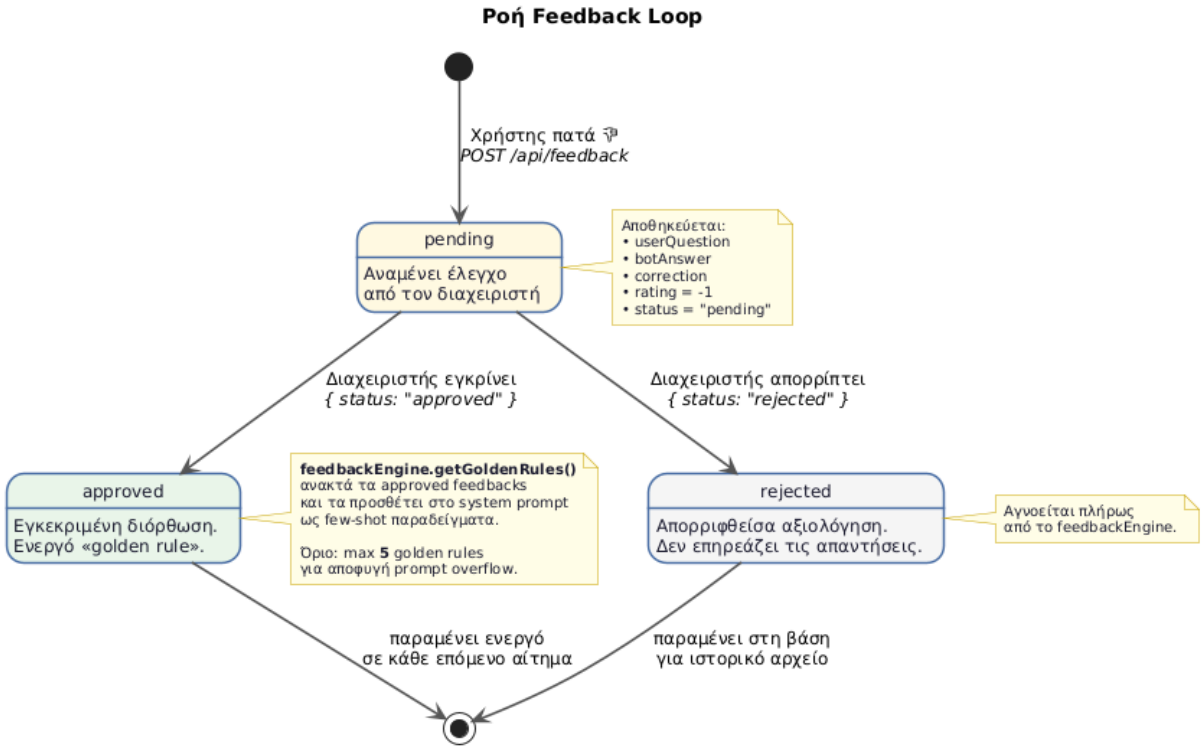
5.5.2 Ροή Feedback Loop (End-to-End)

Η ροή Human-in-the-Loop περιλαμβάνει τέσσερα στάδια που εκτελούνται σε δύο χρονικές στιγμές: άμεσα κατά την αξιολόγηση (στάδια 1–2) και ασύγχρονα κατά το επόμενο αίτημα (στάδια 3–4). Αυτή η ασύγχρονη φύση επιτρέπει στον admin να αξιολογεί feedback στον δικό του ρυθμό, χωρίς να επηρεάζεται η άμεση απόκριση του συστήματος.

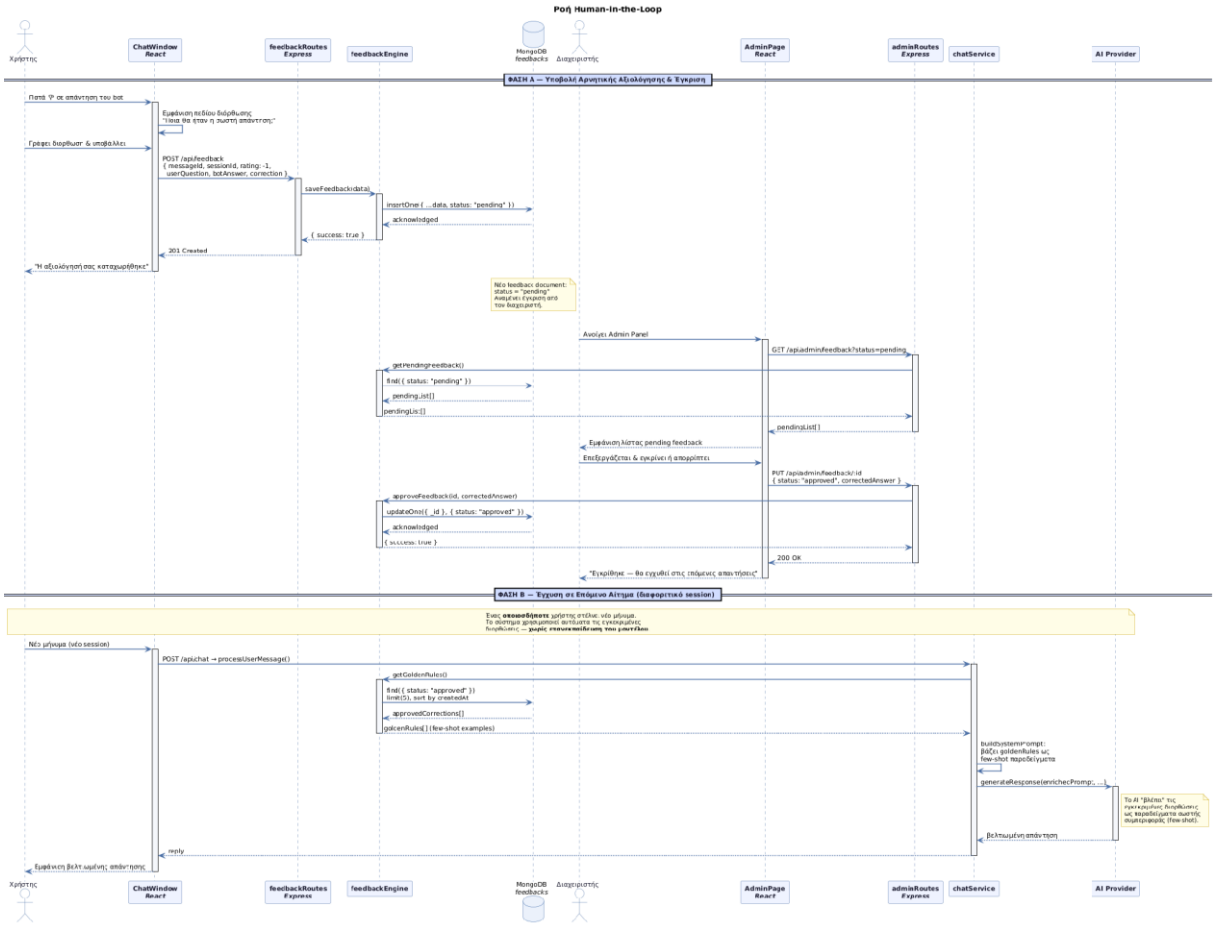
Πίνακας 5.8: Στάδια ροής Human-in-the-Loop

Στάδιο	Actor	Ενέργεια	Αποτέλεσμα στη MongoDB
1. Συλλογή	Χρήστης	Κλικ 👎 + (προαιρετικά) κείμενο διόρθωσης	Νέο feedback document: status="pending"
2. Αναθεώρηση	Admin	Βλέπει pending feedbacks στο admin panel	—
3. Έγκριση	Admin	Κλικ "Approve" (+ επεξεργασία correction αν χρειαστεί)	feedback.status → "approved"
4. Έγχυση	feedbackEngine (αυτόματα)	Σε κάθε νέο chat request, GetGoldenRules() → BuildFeedbackPromptSection()	Χρυσός κανόνας στο system prompt

Πηγή: Ιδία επεξεργασία.



Διάγραμμα 5.4: Feedback loop



Διάγραμμα 5.5: Ροή Human-in-the-loop

5.5.3 Ροή Διαχείρισης Γνωσιακής Βάσης

Η προσθήκη ή επεξεργασία εγγραφής στη γνωσιακή βάση περιλαμβάνει ένα κρίσιμο αυτόματο βήμα: την παραγωγή ή ενημέρωση του embedding. Το σύστημα ανιχνεύει αν άλλαξε το πεδίο content και αυτόματα επανα-εκτελεί την παραγωγή embedding μόνο αν χρειάζεται, αποφεύγοντας περιττές υπολογιστικές κλήσεις για αλλαγές μόνο τίτλου ή κατηγορίας.

Πίνακας 5.9: Λογική αυτόματης ενημέρωσης embedding

Ενέργεια Admin	Αλλάζει το content;	Re-embedding;	Αιτιολόγηση
Προσθήκη νέας εγγραφής	Ναι (νέο)	Πάντα	Νέο content χρειάζεται νέο embedding
Επεξεργασία content	Ναι	Ναι	Νέο νόημα → νέο vector για ορθή αναζήτηση

Ενέργεια Admin	Αλλάζει το content;	Re-embedding;	Αιτιολόγηση
Επεξεργασία μόνο τίτλου	Όχι	Όχι	Το νόημα δεν άλλαξε
Επεξεργασία μόνο κατηγορίας	Όχι	Όχι	Η κατηγορία δεν επηρεάζει vector search
Soft delete (isActive → false)	Όχι	Όχι	Αποκλείεται από isActive filter στη query

Πηγή: Ιδία επεξεργασία.

5.6 Σχεδιασμός Dynamic System Prompt

Ένα από τα πιο σημαντικά αρχιτεκτονικά στοιχεία του συστήματος είναι το δυναμικό system prompt: ένα κείμενο οδηγιών που ανακατασκευάζεται από μηδέν σε κάθε αίτημα chat. Αντί για ένα σταθερό, προκαθορισμένο prompt, το σύστημα συνδυάζει τέσσερα δυναμικά στοιχεία που αλλάζουν ανάλογα με το πλαίσιο.

Πίνακας 5.10: Δομή Dynamic System Prompt

Στρώμα	Πηγή	Ενημερώνεται	Σκοπός
Βασικές οδηγίες	Hardcoded στον κώδικα	Ποτέ (χωρίς deploy)	Ρόλος, γλώσσα, ήθος
RAG Context	MongoDB knowledgebases	Κάθε request (3 πιο σχετικά FAQs)	Εστίαση σε σχετική γνώση
Golden Rules	MongoDB feedbacks (approved)	Κάθε request (έως 5)	Αποφυγή γνωστών λαθών
Negative Examples	MongoDB feedbacks (pending)	Κάθε request (έως 5)	Αποφυγή προβληματικών τύπων απαντήσεων

Πηγή: Ιδία επεξεργασία.

Η διεύθυνση προτεραιότητας είναι ρητή στο σύστημα οδηγιών: οι Golden Rules (εγκεκριμένες διορθώσεις) έχουν απόλυτη προτεραιότητα έναντι του RAG context. Αυτό διασφαλίζει ότι μια διορθωμένη λανθασμένη πληροφορία δεν θα «ξαναεμφανιστεί» ακόμα και αν το αρχικό FAQ δεν έχει ενημερωθεί στη βάση δεδομένων.

5.7 Σύνοψη Κεφαλαίου

Το παρόν κεφάλαιο παρουσίασε τον σχεδιασμό του συστήματος σε επίπεδο τεχνικής λεπτομέρειας. Τα τρία MongoDB schemas (conversations, feedbacks, knowledgebases) σχεδιάστηκαν με κριτήριο τη μείωση joins και την υποστήριξη και των τριών βασικών λειτουργιών: ιστορικό συνομιλίας, Human-in-the-Loop feedback, και semantic search. Τα εννέα API endpoints οργανώνονται σε τρία function groups που αντιστοιχούν στους τρεις ρόλους του συστήματος: χρήστης, admin, και εσωτερική λογική. Το δυναμικό system prompt ενώνει τα τρία στοιχεία (RAG context, διορθώσεις από feedback, βασικές οδηγίες) σε ένα ενιαίο σύστημα οδηγιών που προσαρμόζεται στο εκάστοτε πλαίσιο. Το επόμενο κεφάλαιο παρουσιάζει πώς αυτός ο σχεδιασμός μεταφράστηκε σε κώδικα.

6. Υλοποίηση

Το παρόν κεφάλαιο παρουσιάζει την υλοποίηση των κεντρικών components του ShopEasy ChatBot, με έμφαση στις αποφάσεις που ελήφθησαν κατά την ανάπτυξη και τον τρόπο με τον οποίο αυτές επηρεάζουν τη λειτουργικότητα και την επεκτασιμότητα του συστήματος. Για κάθε component παρουσιάζονται επιλεγμένα αποσπάσματα κώδικα με σχολιασμό, ακολουθούμενα από επεξήγηση των βασικών τεχνικών αποφάσεων. Ο κώδικας είναι σχολιασμένος στα ελληνικά σύμφωνα με τις κατευθυντήριες οδηγίες του έργου, ώστε να είναι κατανοητός σε αναγνώστη χωρίς εξειδικευμένο τεχνικό υπόβαθρο.

6.1 Embedding Service: Τοπική Παραγωγή Διανυσμάτων

Ο embeddingService αποτελεί τη θεμελιώδη υποδομή πάνω στην οποία χτίζεται ολόκληρο το RAG pipeline. Η κεντρική του ευθύνη είναι η μετατροπή κειμένου (ερωτήσεις χρηστών, περιεχόμενο γνωσιακής βάσης) σε αριθμητικά διανύσματα 384 διαστάσεων, χρησιμοποιώντας το τοπικό μοντέλο paraphrase-multilingual-MiniLM-L12-v2 μέσω της βιβλιοθήκης @xenova/transformers.

Η βιβλιοθήκη @xenova/transformers είναι JavaScript port του HuggingFace Transformers, που επιτρέπει την εκτέλεση μοντέλων machine learning απευθείας σε Node.js χωρίς εξαρτήσεις Python. Το μοντέλο φορτώνεται μία φορά κατά την εκκίνηση (warm-up) και επαναχρησιμοποιείται για όλες τις μετέπειτα κλήσεις, αυτό είναι κρίσιμο για την απόδοση, καθώς η φόρτωση ενός neural network (~250MB) διαρκεί 3–5 δευτερόλεπτα.

Για να αποφευχθεί η πολλαπλή φόρτωση του μοντέλου σε concurrent requests, χρησιμοποιείται το Singleton pattern: μια module-level μεταβλητή pipeline κρατά την αναφορά στο φορτωμένο μοντέλο. Εάν δύο requests φτάσουν ταυτόχρονα πριν το μοντέλο φορτωθεί, και οι δύο περιμένουν την ίδια promise αντί να ξεκινήσουν δύο ανεξάρτητες φορτώσεις.

Απόσπασμα Κώδικα 6.1: GetEmbedding – Μετατροπή Κειμένου σε Διάνυσμα (embeddingService.ts)

```
// GetEmbedding: Μετατρέπει κείμενο σε αριθμητικό διάνυσμα 384 διαστάσεων.
// Αν το ίδιο κείμενο έχει ζητηθεί ξανά, επιστρέφει το αποθηκευμένο αποτέλεσμα.
export async function GetEmbedding(text: string): Promise<number[]> {
  // Έλεγχος cache — αποφεύγει επανάληψη υπολογισμού για ίδιο κείμενο
  const cached = embeddingCache.get(text);
  if (cached) return cached;

  const extractor = await GetPipeline(); // Singleton: φορτώνεται μία φορά
  const output = await extractor(text, {
  pooling: 'mean',    // Μέσος όρος token embeddings → ένα διάνυσμα
    normalize: true      // Κανονικοποίηση για cosine similarity
  });
  const embedding: number[] = Array.from(output.data as Float32Array);
  embeddingCache.set(text, embedding); // Αποθήκευση για μελλοντική χρήση
  return embedding;
}
```

Πηγή: Ιδία υλοποίηση, αρχείο backend/src/services/embeddingService.ts

6.1.1 Αποτύπωμα Μοντέλου και Αποθήκευση Docker Volume

Το μοντέλο paraphrase-multilingual-MiniLM-L12-v2 έχει μέγεθος ~250MB και κατεβαίνει αυτόματα από το HuggingFace Hub κατά την πρώτη χρήση. Σε περιβάλλον Docker, χωρίς ρητή διαχείριση, αυτό σημαίνει επανακατέβασμα σε κάθε επανεκκίνηση container,

ανεπίτρεπτο για demo ή ανάπτυξη. Η λύση είναι η δήλωση Docker named volume στο docker-compose.yml που συνδέει τη μόνιμη αποθήκευση του host με τον φάκελο cache του container.

Πίνακας 6.1: Παράμετροι μοντέλου embeddings

Παράμετρος	Τιμή	Σημασία
Μοντέλο	paraphrase-multilingual-MiniLM-L12-v2	HuggingFace identifier
Διαστάσεις embedding	384	Αριθμός float τιμών ανά κείμενο
Υποστηριζόμενες γλώσσες	50+	Συμπεριλαμβάνονται Ελληνικά και Αγγλικά
Cross-lingual matching	Ναι	EN ερώτηση ↔ GR FAQ
Μέγεθος μοντέλου	~250 MB	Κατεβαίνει μία φορά, cached στο Docker volume
Χρόνος warm-up	3–5 δευτερόλεπτα	Μόνο κατά την εκκίνηση
Χρόνος inference	<100ms	Ανά embedding generation

Πηγή: Ιδία επεξεργασία.

6.2 RAG Pipeline: Από την Ερώτηση στο Context

Ο ragService υλοποιεί τη λογική αναζήτησης σχετικών εγγράφων, το «R» (Retrieval) στην αρχιτεκτονική RAG. Αποτελείται από δύο εναλλακτικές υλοποιήσεις που επιλέγονται αυτόματα: η κύρια χρησιμοποιεί το MongoDB Atlas $vectorSearch (semantic search), ενώ η εναλλακτική χρησιμοποιεί το $text search για local development χωρίς Atlas.

6.2.1 Vector Search (Atlas)

Η κύρια μέθοδος αναζήτησης χρησιμοποιεί το aggregation pipeline της MongoDB με τον ειδικό τελεστή $vectorSearch. Η υλοποίηση περιλαμβάνει τέσσερα στάδια pipeline: το $vectorSearch για ANN (Approximate Nearest Neighbor) αναζήτηση στον HNSW index,

το $addFields για προσθήκη του vectorSearchScore ως score πεδίο, το $match για φιλτράρισμα ανενεργών εγγράφων και αποτελεσμάτων κάτω από το κατώφλι ομοιότητας 0.5, και το $project για εξαίρεση του embedding array από τα αποτελέσματα (εξοικονομεί δικτυακό bandwidth).

Σημαντική τεχνική λεπτομέρεια: το $match φίλτρο για isActive και score τοποθετείται εκτός του $vectorSearch operator (ως ξεχωριστό pipeline stage), αντί μέσα στο filter parameter του $vectorSearch. Αυτό αποφεύγει την απαίτηση δήλωσης των πεδίων isActive και score στον Atlas vector index, απλοποιώντας τη διαμόρφωση.

Απόσπασμα Κώδικα 6.2: Aggregation Pipeline $vectorSearch (ragService.ts)

```
// FindRelevantDocsVectorSearch: Σημασιολογική αναζήτηση με MongoDB $vectorSearch.
// Τρέχει 5-στάδιο aggregation pipeline — λειτουργεί μόνο σε MongoDB Atlas.
const results = await KnowledgeBaseModel.aggregate([
  {
    $vectorSearch: {
      index: 'vector_index',      // Όνομα Atlas Search Index
      path: 'embedding',          // Πεδίο που αποθηκεύει τα 384-D vectors
      queryVector: queryEmbedding, // Embedding της ερώτησης χρήστη
      numCandidates: 150,         // Υποψήφια έγγραφα για HNSW αναζήτηση
      limit: limit * 2            // Ζητάμε 2x για να αντισταθμίσουμε φίλτρα
    }
  },
  // Προσθέτουμε το cosine similarity score ως πεδίο
  { $addFields: { score: { $meta: 'vectorSearchScore' } } },
  // Φιλτράρουμε: μόνο ενεργές εγγραφές άνω του κατωφλίου 0.5
  { $match: { isActive: true, score: { $gte: MIN_SIMILARITY_SCORE } } },
  { $limit: limit },
  { $project: { embedding: 0 } } // Εξαιρούμε το μεγάλο array από τα αποτελέσματα
]);
```

Πηγή: Ιδία υλοποίηση, αρχείο backend/src/services/ragService.ts

6.2.2 Παράμετροι Απόδοσης

Πίνακας 6.2: Παράμετροι RAG pipeline

Παράμετρος	Τιμή	Αιτιολόγηση Επιλογής
MAX_RAG_RESULTS (N)	3	Ισορροπία context πληρότητας και token cost
MIN_SIMILARITY_SCORE	0.5	Φιλτράρει άσχετα έγγραφα, επιτρέπει παραφράσεις
VECTOR_SEARCH_CANDIDATES	150	HNSW accuracy/speed trade-off για μικρές βάσεις
Limit query (pre-filter)	N × 2 = 6	Αντισταθμίζει τυχόν inactive εγγραφές μετά το filter

Πηγή: Ιδία επεξεργασία.

6.2.3 Fallback σε Local MongoDB

Για το περιβάλλον ανάπτυξης χωρίς Atlas, ο ragService ανιχνεύει αυτόματα αν η MONGODB_URI ξεκινά με mongodb+srv:// (ένδειξη Atlas σύνδεσης). Εάν όχι, ενεργοποιείται ο FindRelevantDocsTextSearch, που χρησιμοποιεί τον κλασικό $text operator της MongoDB με text index. Η text search δεν κατανοεί νόημα, αναζητά λέξεις-κλειδιά, αλλά είναι επαρκής για τοπική ανάπτυξη και δοκιμές. Το ίδιο RAGResult interface επιστρέφεται και στις δύο περιπτώσεις, διατηρώντας συνεπή API για τον chatService.

6.3 Chat Service: Ενορχήστρωση της Πλήρους Ροής

Ο chatService συντονίζει τα επιμέρους services. Δέχεται το μήνυμα χρήστη, εκτελεί τα έξι βήματα επεξεργασίας με τη σωστή σειρά, και επιστρέφει την απάντηση. Είναι το μοναδικό component που έχει ολική εικόνα της ροής.

6.3.1 Κατασκευή Δυναμικού System Prompt

Το chatService υιοθετεί ασύμμετρη στρατηγική διαχείρισης σφαλμάτων: σφάλματα που δεν εμποδίζουν τη λειτουργία (π.χ. αποτυχία RAG αναζήτησης) αντιμετωπίζονται silently με fallback (επιστροφή κενού array), ενώ κρίσιμα σφάλματα (αποτυχία AI, αδυναμία αποθήκευσης) ανεβαίνουν ως exceptions που αντιμετωπίζει ο centralized error handler του Express. Αυτή η προσέγγιση διασφαλίζει ότι το σύστημα παραμένει λειτουργικό ακόμα και όταν υπο-components αποτυγχάνουν.

Απόσπασμα Κώδικα 6.3: BuildSystemPrompt – Δυναμική Κατασκευή System Prompt (chatService.ts)

```
// BuildSystemPrompt: Κατασκευάζει δυναμικά το system prompt για
κάθε request.
// Αλλάζει σε κάθε κλήση ανάλογα με το RAG context και τα
εγκεκριμένα feedbacks.
```

```
async function BuildSystemPrompt(ragContext: string):
Promise<string> {
  const baseInstructions =
    'Είσαι ο ψηφιακός βοηθός υποστήριξης του ShopEasy. ' +
    'Απάντα μόνο με βάση τις πληροφορίες που σου δίνονται.';

  // Ενότητα RAG: τα πιο σχετικά FAQs ή fallback μήνυμα
  const knowledgeSection = ragContext
    ? `ΠΛΗΡΟΦΟΡΙΕΣ ΑΠΟ ΤΗ ΓΝΩΣΙΑΚΗ ΒΑΣΗ:\n${ragContext}`
    : 'ΠΛΗΡΟΦΟΡΙΕΣ: Δεν βρέθηκαν σχετικές πληροφορίες.';

  // Παράλληλη ανάκτηση golden rules και αρνητικών παραδειγμάτων
  const [goldenRules, negativeExamples] = await Promise.all([
    GetGoldenRules(), GetNegativeExamples()
  ]);
  const feedbackSection =
    BuildFeedbackPromptSection(goldenRules, negativeExamples);

  // Ενσωμάτωση feedback μόνο αν υπάρχει — αποφεύγει φούσκωμα prompt
  return feedbackSection
    ?
`${baseInstructions}\n\n${knowledgeSection}\n\n${feedbackSection}`
    : `${baseInstructions}\n\n${knowledgeSection}`;
}
```

Πηγή: Ιδία υλοποίηση, αρχείο backend/src/services/chatService.ts

6.4 Feedback Engine: Inference-Time Alignment

Ο feedbackEngine υλοποιεί τη λογική του Human-in-the-Loop μηχανισμού από την πλευρά του AI: μετατρέπει τις αποθηκευμένες διορθώσεις σε κείμενο κατάλληλο για εισαγωγή στο system prompt. Αποτελεί τη "μνήμη" του bot, χωρίς επανεκπαίδευση.

6.4.1 GetGoldenRules και GetNegativeExamples

Η GetGoldenRules ανακτά feedbacks με rating: -1 και status: "approved" (εγκεκριμένες διορθώσεις), ταξινομημένα από νεότερα προς παλαιότερα, με όριο έως 5 εγγραφές. Η επιλογή "νεότερα πρώτα" έχει λογική αιτία: πιο πρόσφατες διορθώσεις αντικατοπτρίζουν την πιο ενημερωμένη πολιτική της επιχείρησης. Η GetNegativeExamples ακολουθεί ίδια λογική αλλά για status: "pending" εγγραφές, αυτές χρησιμοποιούνται ως αποτρεπτικά παραδείγματα ακόμα και πριν την έγκριση.

Απόσπασμα Κώδικα 6.4: GetGoldenRules – Ανάκτηση Εγκεκριμένων Διορθώσεων (feedbackEngine.ts)

```
// GetGoldenRules: Ανακτά εγκεκριμένες διορθώσεις από τη MongoDB.
```

```
// Αποτέλεσμα: έως MAX_GOLDEN_RULES (=5) εγγραφές, νεότερες πρώτα.
export async function GetGoldenRules(
    limit: number = MAX_GOLDEN_RULES) {
  return FeedbackModel
    .find({
      rating: -1,              // Μόνο αρνητικές αξιολογήσεις (👎)
      status: 'approved',      // Μόνο εγκεκριμένες από admin
      correction: { $ne: null } // Υποχρεωτικά με διόρθωση
    })
    .sort({ createdAt: -1 }) // Νεότερες πρώτα — πιο ενημερωμένη
πολιτική
    .limit(limit)            // Cap: αποτρέπει φούσκωμα system
prompt
    .lean();                 // Απλά JS objects, χωρίς Mongoose
overhead
}
```

Πηγή: Ιδία υλοποίηση, αρχείο backend/src/services/feedbackEngine.ts

6.4.2 Μορφοποίηση BuildFeedbackPromptSection

Η BuildFeedbackPromptSection παράγει δύο τμήματα κειμένου: την ενότητα golden rules με τη μορφή «Ε: [ερώτηση] | ❌ Παλιά απάντηση: [...] | ✅ ΣΩΣΤΗ απάντηση: [...]» και την ενότητα negative examples με απλά σημεία. Η χρήση των emoji ✅/❌ έχει λειτουργικό σκοπό και όχι αισθητικό. Στις δοκιμές που έγιναν με και χωρίς αυτά τα σύμβολα, το μοντέλο φάνηκε να ακολουθεί πιο σταθερά τους κανόνες όταν τα κρίσιμα σημεία ξεχώριζαν οπτικά μέσα στο context.

Πίνακας 6.3: Μορφή golden rule στο system prompt

Στοιχείο	Πηγή	Ρόλος
Ε: "[userQuestion]"	feedback.userQuestion	Πλαίσιο: σε ποια ερώτηση αναφέρεται ο κανόνας
❌ Παλιά απάντηση: "[botAnswer]"	feedback.botAnswer	Τι να ΜΗΝ επαναλάβει το bot
✅ ΣΩΣΤΗ απάντηση: "[correction]"	feedback.correction	Τι ακριβώς να πει — υψηλότερη προτεραιότητα

Πηγή: Ιδία υλοποίηση, αρχείο feedbackEngine.ts

6.5 Αφαίρεση Παρόχου AI (Factory Pattern)

Ένα από τα πιο χρήσιμα τεχνικά στοιχεία της υλοποίησης είναι η αρχιτεκτονική αφαίρεσης παρόχου AI. Αντί να χρησιμοποιεί απευθείας το SDK ενός παρόχου, το σύστημα ορίζει ένα TypeScript interface (AIProvider) με μία μόνο μέθοδο: GenerateResponse(messages,

systemPrompt). Κάθε πάροχος υλοποιεί το interface χρησιμοποιώντας το αντίστοιχο SDK του, ενώ ο chatService καλεί αποκλειστικά το interface χωρίς να γνωρίζει τον υποκείμενο πάροχο.

6.5.1 Το AIProvider Interface

Το interface ορίζει ένα συμβόλαιο (contract) μεταξύ του chatService και των παρόχων: οποιοσδήποτε πάροχος που υλοποιεί αυτό το interface είναι εναλλάξιμος χωρίς αλλαγή κώδικα. Αυτή η αρχή, γνωστή ως Dependency Inversion Principle (DIP), ένα από τα SOLID principles, έχει άμεση ερευνητική αξία: επιτρέπει τη σύγκριση ποιότητας απαντήσεων μεταξύ διαφορετικών μοντέλων (Gemini vs GPT-4o vs Llama) υπό ακριβώς τις ίδιες συνθήκες, αλλάζοντας μόνο μία γραμμή στο .env αρχείο.

6.5.2 Διαχείριση Ιστορικού Συνομιλίας Ανά Πάροχο

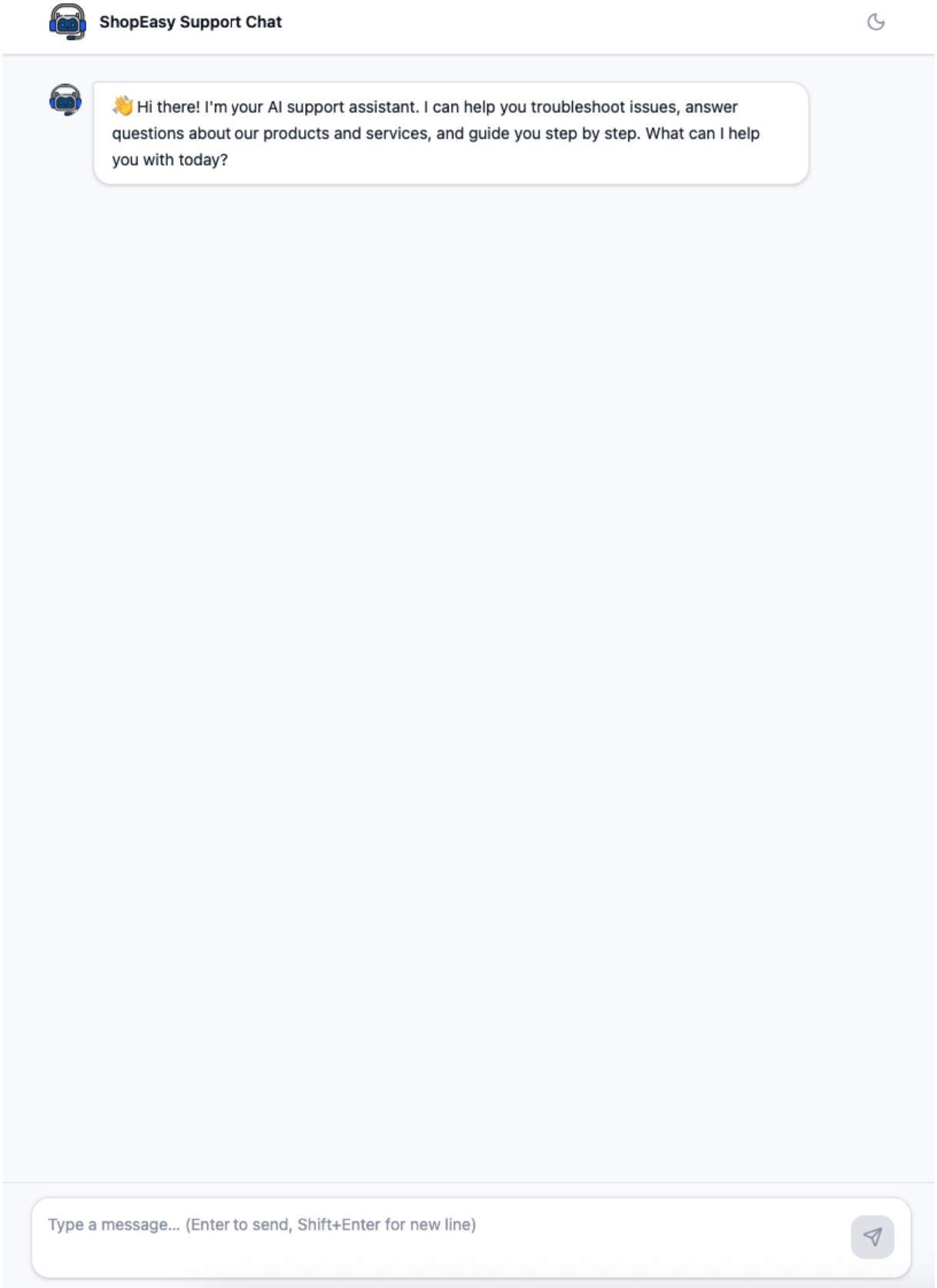
Η μετατροπή του εσωτερικού format μηνυμάτων (Message interface) στο format που απαιτεί κάθε πάροχος αποτελεί ευθύνη κάθε AIProvider υλοποίησης. Το OpenAI και OpenRouter αναμένουν messages ως πίνακα { role, content } objects, format ταυτόσημο με το εσωτερικό του συστήματος, άρα δεν απαιτείται μετατροπή. Το Google Gemini χρησιμοποιεί διαφορετικό format (parts array, ξεχωριστό systemInstruction) και διαφορετική ορολογία ("model" αντί "assistant"), αυτή η διαφορά αποκρύπτεται πλήρως μέσα στη GeminiProvider κλάση.

Πίνακας 6.4: Σύγκριση format μηνυμάτων ανά πάροχο

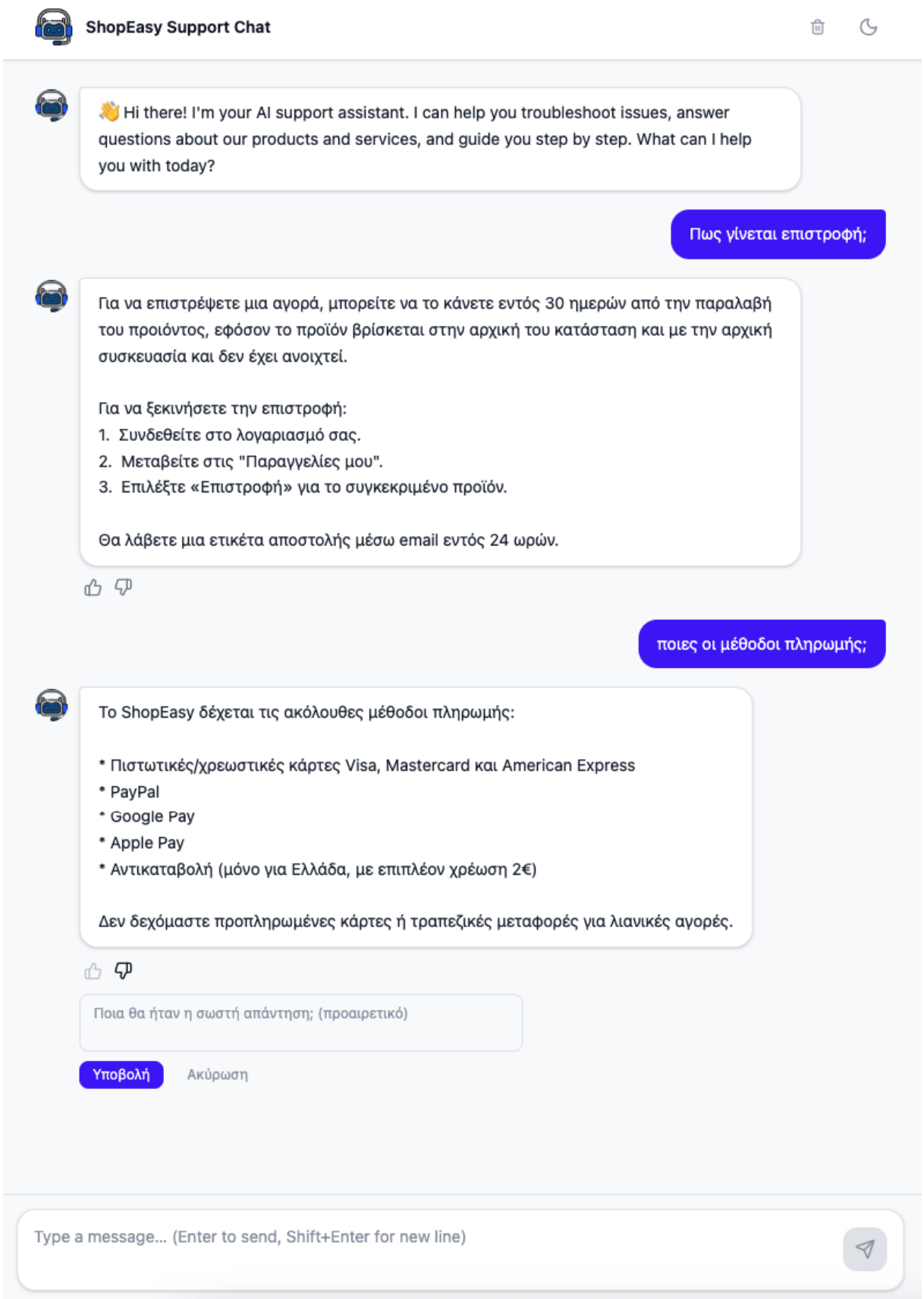
Πάροχος	Role ορολογία	System prompt	Format μηνυμάτων
OpenAI	"user" / "assistant"	Ως { role: "system" } μήνυμα	[{ role, content }]
OpenRouter	"user" / "assistant"	Ως { role: "system" } μήνυμα	[{ role, content }]
Google Gemini	"user" / "model"	Ξεχωριστό systemInstruction πεδίο	[{ role, parts: [{ text }] }]

*Πηγή: Τεχνική τεκμηρίωση παρόχων, αρχεία *Provider.ts.*

6.6 Chat Interface: Frontend Υλοποίηση



Εικόνα 6.1: Αρχική οθόνη



Εικόνα 6.2: Παραδείγματα ερωτήσεων

Το frontend του συστήματος αποτελείται από τέσσερα React components που διαχωρίζουν σαφώς τις ευθύνες: το ChatWindow (κεντρικό state management), το MessageBubble (εμφάνιση μηνύματος), το FeedbackButtons (αλληλεπίδραση αξιολόγησης) και το AdminPage (admin panel).

6.6.1 Session Management και Persistence

Το ChatWindow διαχειρίζεται δύο τύπους persistence. Για την ταυτότητα της συνομιλίας (sessionId), χρησιμοποιεί localStorage: κατά την αρχικοποίηση ελέγχει αν υπάρχει αποθηκευμένο sessionId και το επαναχρησιμοποιεί, διαφορετικά δημιουργεί νέο μέσω crypto.randomUUID(). Αυτό εξασφαλίζει ότι μια συνομιλία δεν διακόπτεται από ανανέωση σελίδας. Για το ιστορικό μηνυμάτων, το πλήρες messages array serialized ως JSON αποθηκεύεται στο localStorage μετά από κάθε αλλαγή, χρησιμοποιώντας useEffect hook.

Σημαντική διευκρίνιση: το localStorage persistence είναι αποκλειστικά για το UI, το backend αποθηκεύει πάντα το πλήρες ιστορικό στη MongoDB ανεξάρτητα. Αν ο χρήστης καθαρίσει το localStorage ή χρησιμοποιήσει διαφορετικό browser, η οθόνη θα εμφανίζεται κενή, αλλά το backend ιστορικό παραμένει άθικτο και το AI εξακολουθεί να έχει πρόσβαση σε αυτό (μέσω sessionId).

6.6.2 Two-Step Clear και Error Banner

Δύο UI μηχανισμοί αξίζουν ιδιαίτερης αναφοράς λόγω της αρχιτεκτονικής τους. Ο πρώτος είναι το two-step clear: κατά την εκκαθάριση ιστορικού, η πρώτη πίεση του κουμπιού αλλάζει μόνο το state confirmingClear σε true (εμφανίζοντας "Clear?"), ενώ μόνο η δεύτερη πίεση εκτελεί την πραγματική εκκαθάριση. Εάν ο χρήστης δεν επιβεβαιώσει εντός 3 δευτερολέπτων (CLEAR_CONFIRM_TIMEOUT_MS), το κουμπί επιστρέφει αυτόματα στην αρχική κατάσταση. Ο δεύτερος είναι το error banner, το οποίο αυτοαπορρίπτεται μετά από 4 δευτερόλεπτα (ERROR_AUTO_DISMISS_MS) μέσω setTimeout· το timer ID αποθηκεύεται σε useRef για ακύρωση αν εμφανιστεί νέο σφάλμα πριν λήξει ο χρόνος.

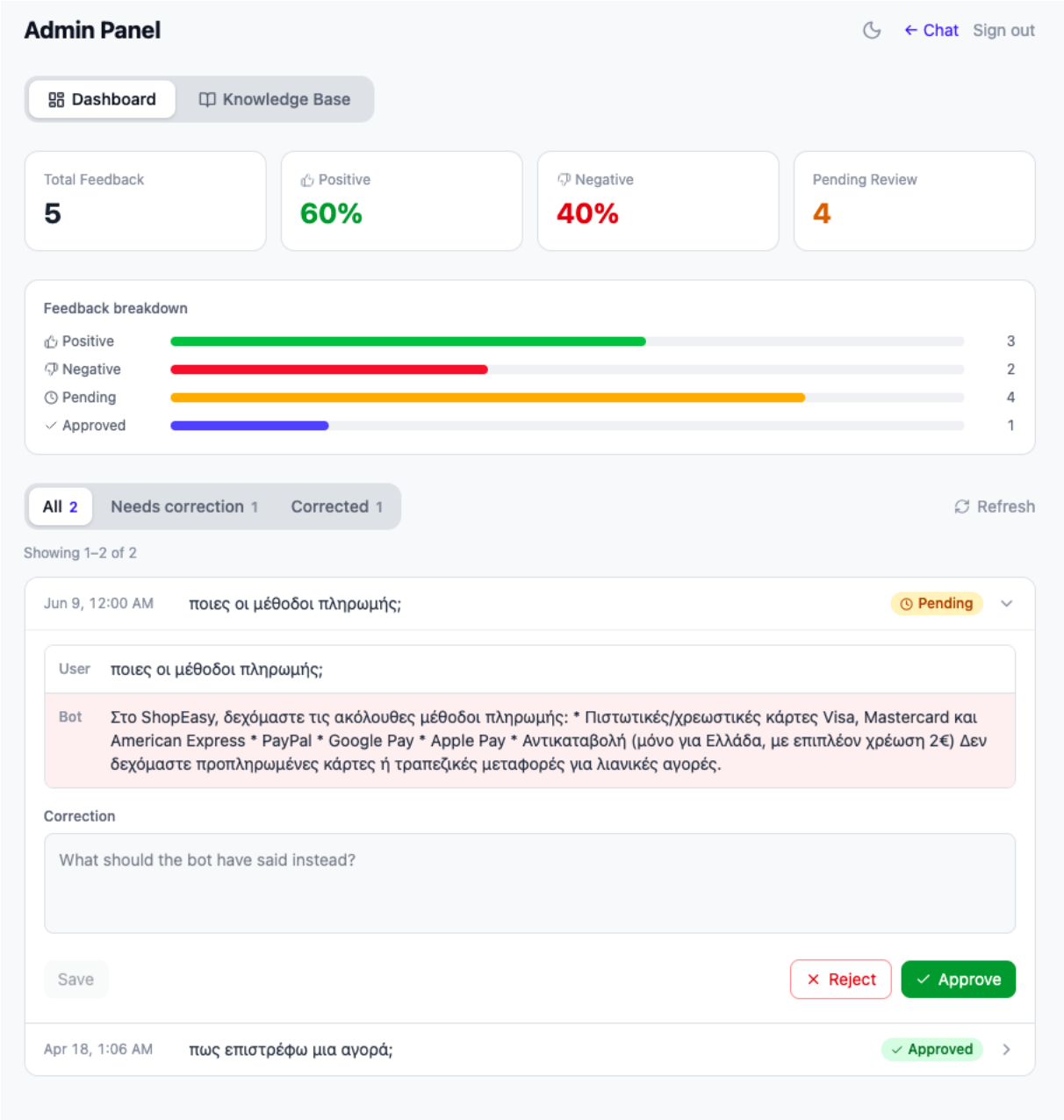
6.6.3 FeedbackButtons: Κλιμακωτή Υποβολή

Το FeedbackButtons component υλοποιεί διαφορετική συμπεριφορά για τις δύο ψήφους. Η θετική (👍) υποβάλλεται αμέσως μετά το κλικ. Η αρνητική (👎) εμφανίζει πρώτα ένα πεδίο κειμένου "Ποια θα ήταν η σωστή απάντηση;", ο χρήστης μπορεί να γράψει διόρθωση ή να υποβάλει χωρίς κείμενο. Η οριστική υποβολή γίνεται μόνο μετά το κλικ "Υποβολή". Αυτή

η κλιμακωτή (progressive disclosure) προσέγγιση έχει δύο πλεονεκτήματα: αποφεύγει τυχαίες αρνητικές ψήφους και αυξάνει την πιθανότητα λήψης ποιοτικής διόρθωσης.

Μετά την υποβολή, το state voted κλειδώνει αμφότερα τα κουμπιά και επισημαίνει οπτικά το επιλεγμένο. Αυτό αποτρέπει διπλή υποβολή (idempotency) και παρέχει στον χρήστη οπτική επιβεβαίωση ότι η ψήφος καταγράφηκε.

6.7 Admin Panel: Πλήρης Κύκλος Διαχείρισης



Εικόνα 6.3: Admin panel (dashboard)

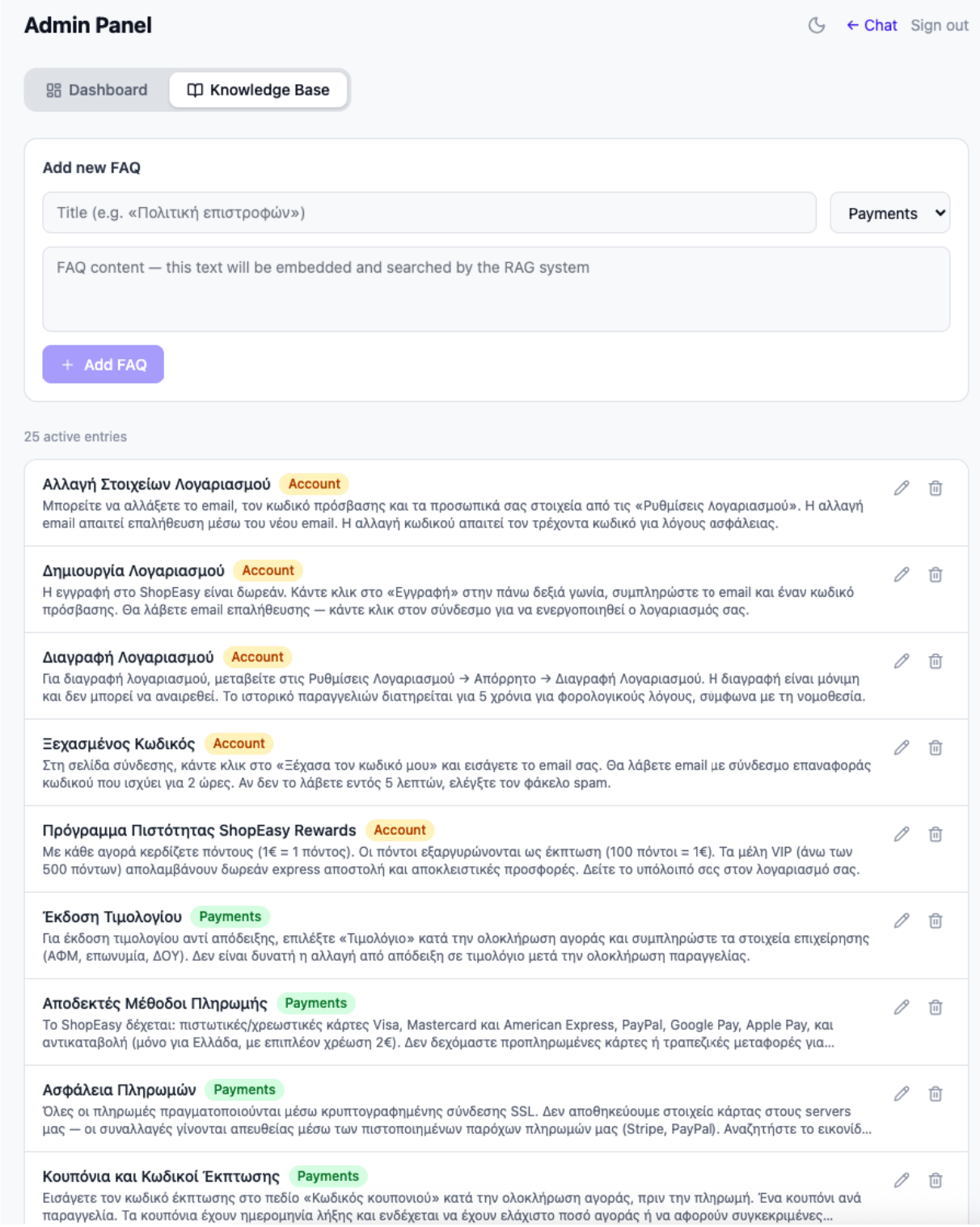
Το AdminPage component υλοποιεί τον πίνακα διαχείρισης σε μία σελίδα (single-page) με δύο κύριες καρτέλες: Dashboard (διαχείριση feedback) και Knowledge Base (διαχείριση γνωσιακής βάσης). Η πρόσβαση γίνεται με απλή password authentication (demo credentials: admin/admin), επαρκής για ακαδημαϊκό πρωτότυπο, με σαφή σημείωση ότι σε παραγωγικό σύστημα θα αντικαθίστατο από bcrypt + JWT.

6.7.1 Dashboard Tab: Workflow Εγκρίσεων

Η Dashboard καρτέλα εμφανίζει τέσσερις κάρτες στατιστικών (σύνολο feedback, % θετικών, % αρνητικών, εκκρεμή) που ενημερώνονται optimistically κατά την έγκριση/απόρριψη, χωρίς αναφόρτωση σελίδας. Κάτω από τις κάρτες εμφανίζεται ένα CSS-only bar chart που απεικονίζει την κατανομή κατά κατάσταση (pending/approved/rejected). Αξίζει να σημειωθεί ότι αντί για chart library (Chart.js, Recharts), χρησιμοποιείται απλό Tailwind CSS με δυναμικά widths, αρκετά για το σκοπό αυτό και χωρίς πρόσθετες εξαρτήσεις.

Κάθε feedback εγγραφή εμφανίζεται σε expandable κάρτα που δείχνει την ερώτηση του χρήστη, την απάντηση του bot (με κόκκινο highlight), και ένα πεδίο επεξεργασίας για τη διόρθωση. Ο admin μπορεί να τροποποιήσει τη διόρθωση πριν εγκρίνει. Τα κουμπιά "Approve" / "Reject" στέλνουν PUT /api/admin/feedback/:messageId με το κατάλληλο status.

6.7.2 Knowledge Base Tab: Live Embedding Updates



Εικόνα 6.4: Admin Panel (Knowledge base tab)

Η Knowledge Base καρτέλα παρουσιάζει λίστα όλων των ενεργών εγγραφών με color-coded category badges. Η φόρμα προσθήκης νέας εγγραφής περιλαμβάνει title, content textarea, και category dropdown. Κατά την υποβολή, το POST /api/admin/knowledge endpoint

παράγει αυτόματα το embedding στο backend, ο admin δεν χρειάζεται να γνωρίζει τίποτα για vectors ή machine learning.

Η επεξεργασία γίνεται με inline form: το κλικ "Edit" μετατρέπει τη στατική προβολή της εγγραφής σε form με προ-συμπληρωμένα πεδία. Κατά την αποθήκευση, ο backend ανιχνεύει αν άλλαξε το content και παράγει νέο embedding μόνο αν χρειάζεται. Η διαγραφή χρησιμοποιεί two-step confirmation (παρόμοιο με το clear chat): πρώτο κλικ εμφανίζει "Confirm?", δεύτερο εκτελεί soft delete.

6.8 Seeding Γνωσιακής Βάσης

Το script seedKnowledge.ts είναι ένα one-time εκτελέσιμο script που δημιουργεί τις 25 αρχικές εγγραφές FAQ στη βάση δεδομένων. Η εκτέλεσή του γίνεται μία φορά μετά την πρώτη εκκίνηση του συστήματος. Το script χρησιμοποιεί idempotency check, ελέγχει αν η βάση δεδομένων είναι ήδη γεμάτη και αν ναι, τερματίζει χωρίς να κάνει αλλαγές, αποτρέποντας διπλοεγγραφές σε επαναλαμβανόμενες εκτελέσεις.

Πίνακας 6.5: Κατανομή 25 FAQ εγγραφών ανά κατηγορία

Κατηγορία	Πλήθος	Παραδείγματα Θεμάτων
returns	5	Πολιτική 30 ημερών, χρόνος επιστροφής χρημάτων (5–7 ημέρες), ανταλλαγή, ελαττωματικά, μη επιστρέψιμα
shipping	5	Χρόνος παράδοσης (3–5 ημέρες), δωρεάν αποστολή (άνω €39), tracking, εναλλακτική διεύθυνση, αποτυχημένη παράδοση
payments	5	Μέθοδοι πληρωμής (Visa/PayPal/Apple Pay), ασφάλεια (SSL/Stripe), τιμολόγιο, πρόβλημα χρέωσης, κωδικοί έκπτωσης
products	5	Δείκτες διαθεσιμότητας, οδηγός μεγεθών, αυθεντικότητα, σύγκριση προϊόντων, πολιτική κριτικών
account	5	Δημιουργία λογαριασμού, λησμονημένος κωδικός, αλλαγή στοιχείων, διαγραφή λογαριασμού, ShopEasy Rewards

Πηγή: Ιδία υλοποίηση, αρχείο seedKnowledge.ts

Όλες οι εγγραφές είναι γραμμένες στα ελληνικά. Το multilingual embedding μοντέλο διασφαλίζει cross-lingual αναζήτηση: ερωτήσεις στα αγγλικά (π.χ. "how to return an item") αντιστοιχίζονται σωστά με τις ελληνικές εγγραφές χωρίς ρητή μετάφραση.

6.9 Infrastructure: Docker Compose

Η υποδομή εκτέλεσης ορίζεται στο docker-compose.yml που εκκινεί τρεις υπηρεσίες: το backend container (Node.js/Express), τη MongoDB container (local development database),

και το Mongo Express container (web UI για direct database inspection). Ο backend container συνδέεται με ένα named volume (model_cache) που διατηρεί το κατεβασμένο embedding model μεταξύ επανεκκινήσεων.

Πίνακας 6.6: Docker Compose υπηρεσίες

Υπηρεσία	Image	Port	Σκοπός
backend	Custom Node.js image	3000:3000	Express API server + embedding model
mongodb	mongo:7	27017:27017	Τοπική βάση δεδομένων (development)
mongo-express	mongo-express	8081:8081	Web UI για διαχείριση MongoDB (admin/admin)

Πηγή: Ιδία υλοποίηση, αρχείο docker-compose.yml

6.10 Τεχνικές Προκλήσεις και Εμπειρίες Υλοποίησης

Latency του Embedding Service: Πρώτη Εκτέλεση και Warm-Up. Το πιο εμφανές πρόβλημα κατά την ανάπτυξη ήταν ο χρόνος αρχικοποίησης του embedding service. Κατά την πρώτη εκτέλεση, το @xenova/transformers χρειάζεται να φορτώσει το μοντέλο paraphrase-multilingual-MiniLM-L12-v2 (~250MB) και να το μεταφράσει στη μορφή ONNX. Αυτό προκαλούσε latency 8-12 δευτερολέπτων στο πρώτο request μετά την εκκίνηση του container, κάτι που σε demo περιβάλλον γίνεται εμφανώς αντιληπτό. Η λύση ήταν ο συνδυασμός Docker volume caching (ώστε το μοντέλο να μην κατεβαίνει ξανά σε κάθε restart) και ένα warm-up request κατά την εκκίνηση του backend, ώστε το μοντέλο να είναι ήδη φορτωμένο πριν φτάσει το πρώτο πραγματικό request.

Ασυμφωνία Embeddings μετά από Επεξεργασία Εγγραφής (Embedding Mismatch). Ένα λεπτό αλλά κρίσιμο πρόβλημα εντοπίστηκε κατά τις πρώτες δοκιμές του admin panel: όταν ο διαχειριστής επεξεργαζόταν το content μιας εγγραφής της γνωσιακής βάσης χωρίς να γίνει αυτόματα νέο embedding, το σύστημα εξακολουθούσε να χρησιμοποιεί το παλιό διάνυσμα για την αναζήτηση. Αποτέλεσμα: ερωτήσεις που θεωρητικά έπρεπε να ταιριάζουν με την ενημερωμένη εγγραφή επέστρεφαν είτε λάθος αποτέλεσμα είτε τίποτα. Η αντιμετώπιση έγινε στο admin endpoint PUT /api/admin/knowledge/:id, που πλέον καλεί πάντα embeddingService.getEmbedding() και ενημερώνει το πεδίο embedding πριν αποθηκεύσει.

Ρύθμιση MIN_SIMILARITY_SCORE: Ισορροπία μεταξύ Precision και Recall. Η αρχική τιμή του MIN_SIMILARITY_SCORE ήταν 0.7, τιμή που φάνηκε πολύ αυστηρή στην πράξη. Σε αρκετά εύλογα ερωτήματα χρηστών (π.χ. "πότε έρχεται η παραγγελία μου;") το RAG δεν επέστρεφε κανένα αποτέλεσμα, με αποτέλεσμα το AI να απαντά αποκλειστικά από τις γενικές του γνώσεις χωρίς specific context από το κατάστημα. Μετά από δοκιμές με διαφορετικές ερωτήσεις, η τιμή χαμηλώσε στο 0.5, που δίνει καλύτερο recall χωρίς να επιστρέφει άσχετα αποτελέσματα. Η παράμετρος παραμένει εκτεθειμένη ως named constant ακριβώς για να επιτρέπει εύκολη μελλοντική ρύθμιση.

Overflow του System Prompt και Cap στα Golden Rules. Κατά τη δοκιμή του feedback loop, διαπιστώθηκε ότι η σωρευτική προσθήκη εγκεκριμένων διορθώσεων στο system prompt μπορεί γρήγορα να οδηγήσει σε token overflow ή σε μείωση της ποιότητας των απαντήσεων λόγω διάσπασης της προσοχής του μοντέλου. Με 10+ golden rules, το μοντέλο ξεκινούσε να "αγνοεί" μέρος των οδηγιών. Αυτό οδήγησε στην εισαγωγή των σταθερών MAX_GOLDEN_RULES=5 και MAX_NEGATIVE_EXAMPLES=5 στον feedbackEngine, που διατηρούν το prompt σε διαχειρίσιμο μέγεθος επιλέγοντας τις πιο πρόσφατες εγγραφές.

Fallback στο $text Search: Προβλήματα και Περιορισμοί. Το fallback από $vectorSearch σε $text search για τοπική ανάπτυξη λειτούργησε αλλά με αξιοσημείωτη διαφορά στην ποιότητα αποτελεσμάτων. Το $text search της MongoDB βασίζεται σε keyword matching, οπότε σημασιολογικά παρόμοια ερωτήματα που δεν μοιράζονται ακριβείς λέξεις με τις εγγραφές δεν επιστρέφουν αποτέλεσμα. Για παράδειγμα, "θέλω να επιστρέψω ένα προϊόν" δεν ταίριαζε με εγγραφές που περιείχαν "πολιτική επιστροφών" χωρίς vector search. Αυτός ο περιορισμός τεκμηριώθηκε ρητά στο αρχείο ragService.ts και αποτελεί ένα από τα βασικά επιχειρήματα για χρήση MongoDB Atlas αντί local instance σε παραγωγικό περιβάλλον.

6.11 Σύνοψη Κεφαλαίου

Το κεφάλαιο κάλυψε την υλοποίηση των βασικών components του συστήματος. Ο embeddingService διασφαλίζει παραγωγή διανυσμάτων χωρίς cloud API, με Singleton pattern για αποδοτική φόρτωση μοντέλου. Ο ragService υλοποιεί σημασιολογική αναζήτηση με αυτόματο fallback για συμβατότητα μεταξύ Atlas και local περιβάλλοντος. Ο chatService ενορχηστρώνει τα έξι βήματα επεξεργασίας με δυναμικό system prompt που ανακατασκευάζεται σε κάθε request. Ο feedbackEngine μετατρέπει ανθρώπινες διορθώσεις σε ευθυγράμμιση κατά την εκτέλεση χωρίς επανεκπαίδευση. Το frontend ChatWindow

διαχειρίζεται session persistence, notification sounds, και progressive disclosure feedback UX. Το AdminPage παρέχει πλήρη workflow εγκρίσεων και live embedding updates. Στο επόμενο κεφάλαιο αξιολογείται πώς λειτουργεί αυτό το σύστημα στην πράξη.

7. Αξιολόγηση

Το παρόν κεφάλαιο παρουσιάζει την ποσοτική και ποιοτική αξιολόγηση του ShopEasy ChatBot. Η αξιολόγηση δομείται γύρω από τρία ανεξάρτητα πειράματα, καθένα από τα οποία στοχεύει σε διαφορετική διάσταση του συστήματος: την αποτελεσματικότητα του μηχανισμού ανάκτησης πληροφορίας (RAG), τη συνολική ακρίβεια του συστήματος σε σύγκριση με ένα baseline χωρίς RAG, και την πρακτική επίδραση του Human-in-the-Loop μηχανισμού ανατροφοδότησης. Όλα τα πειράματα εκτελέστηκαν την 1η Ιουνίου 2026, με τις παραμέτρους του συστήματος σε production-equivalent ρυθμίσεις.

7.1 Μεθοδολογία Αξιολόγησης

Η αξιολόγηση σχεδιάστηκε με βάση την αρχή της ablation study (μελέτη αφαίρεσης), μια καθιερωμένη πρακτική στον τομέα της Επεξεργασίας Φυσικής Γλώσσας: τα ευρήματα ενός συστήματος είναι αξιόπιστα μόνο όταν συγκρίνονται με ένα baseline στο οποίο έχει αφαιρεθεί το υπό μελέτη χαρακτηριστικό. Συγκεκριμένα, η αποτελεσματικότητα του RAG αξιολογείται μέσω παράλληλης εκτέλεσης δύο εκδοχών του συστήματος: της πλήρους εκδοχής (full-chat, με RAG ενεργό) και ενός baseline (μόνο γλωσσικό μοντέλο χωρίς γνωσιακή βάση).

Για την αξιοπιστία και την επαναληψιμότητα των πειραμάτων ελήφθησαν τα ακόλουθα μέτρα. Πρώτον, όλες οι κλήσεις προς το γλωσσικό μοντέλο εκτελέστηκαν με παράμετρο temperature = 0, εξασφαλίζοντας ντετερμινιστικές απαντήσεις: η ίδια ερώτηση δίνει πάντοτε την ίδια απάντηση. Δεύτερον, οι συνομιλίες των πειραμάτων αποθηκεύονται με μοναδικό πρόθεμα συνεδρίας (EVAL_) ώστε να διαχωρίζονται από πραγματικά δεδομένα και να είναι δυνατός ο πλήρης καθαρισμός μετά τις δοκιμές. Τρίτον, όλες οι ερωτήσεις και τα αποτελέσματα διατηρούνται σε αρχεία JSON και CSV για μετέπειτα επανέλεγχο.

Το γλωσσικό μοντέλο που χρησιμοποιήθηκε σε όλα τα πειράματα είναι το llama-3.3-70b-versatile (LLaMA 3.3 με 70 δισεκατομμύρια παραμέτρους) μέσω του Groq API. Η επιλογή του Groq έγινε για πρακτικούς λόγους που συζητούνται στην ενότητα 7.6: τα δωρεάν επίπεδα των

Gemini (20 αιτήματα ανά ημέρα) και OpenRouter (συστηματικές υπερφορτώσεις του παρόχου Venice) αποδείχθηκαν ανεπαρκή για μια ολοκληρωμένη αξιολόγηση 30 ερωτήσεων. Το Groq προσφέρει 14.400 αιτήματα ημερησίως δωρεάν και εξυπηρετεί το ίδιο μοντέλο LLaMA 3.3-70b με ντετερμινιστικά αποτελέσματα.

Το embedding μοντέλο παρέμεινε σε όλα τα πειράματα το paraphrase-multilingual-MiniLM-L12-v2 (384 διαστάσεων, πολυγλωσσικό), όπως τεκμηριώνεται στο Κεφάλαιο 4.

7.2 Σχεδιασμός Test Set

Για την αξιολόγηση κατασκευάστηκε σύνολο ελέγχου 30 ερωτήσεων, οργανωμένων σε τέσσερις κατηγορίες οι οποίες αντιστοιχούν στους κύριους τύπους ερωτημάτων που λαμβάνει ένα σύστημα εξυπηρέτησης πελατών σε πραγματικές συνθήκες. Το μέγεθος του δείγματος (30 ερωτήσεις) επιλέχθηκε ως συμβιβασμός μεταξύ στατιστικής αξιοπιστίας και πρακτικής εφικτότητας του χειροκίνητου ελέγχου των απαντήσεων.

7.2.1 Οι Τέσσερις Κατηγορίες Ερωτήσεων

Άμεσες ερωτήσεις (Direct, D01–D10): Δέκα ερωτήσεις διατυπωμένες με τρόπο πολύ κοντινό στους τίτλους των FAQ της γνωσιακής βάσης. Αποτελούν τις «εύκολες» περιπτώσεις: αν το σύστημα αποτυγχάνει εδώ, υπάρχει θεμελιώδες πρόβλημα στη λειτουργία του. Καλύπτουν και τις πέντε κατηγορίες FAQ (επιστροφές, αποστολή, πληρωμές, προϊόντα, λογαριασμός) με δύο ερωτήσεις ανά κατηγορία.

Παραφρασμένες ερωτήσεις (Paraphrased, P01–P10): Δέκα ερωτήσεις που στοχεύουν στις ίδιες δέκα FAQ με την κατηγορία Direct, αλλά διατυπωμένες με εντελώς διαφορετικές λέξεις. Παράδειγμα: στη θέση της D01 «Πόσες μέρες έχω για να επιστρέψω ένα προϊόν;» υπάρχει η P01 «Μπορώ να στείλω πίσω κάτι που αγόρασα;». Η κατηγορία αυτή είναι κρίσιμη: ένα σύστημα βασισμένο σε αναζήτηση λέξεων-κλειδιών θα αποτύγχανε σε πολλές, ενώ μόνο η σημασιολογική αναζήτηση μπορεί να τις αντιμετωπίσει.

Άσχετες ερωτήσεις (Off-topic, O01–O05): Πέντε ερωτήσεις εντελώς εκτός θέματος (π.χ. ο πρωθυπουργός της Ελλάδας, μια συνταγή σπαγγέτι, ένα ποίημα). Το σύστημα οφείλει να μην επιστρέφει σχετικά έγγραφα και ο τελικός βοηθός να αρνείται ευγενικά να απαντήσει. Αυτή η κατηγορία αξιολογεί την ικανότητα του συστήματος να αναγνωρίζει τα όρια της αρμοδιότητάς του.

Διαγλωσσικές ερωτήσεις (Cross-lingual, X01–X05): Πέντε ερωτήσεις στα Αγγλικά που στοχεύουν σε FAQ γραμμένες στα Ελληνικά. Αξιολογούν άμεσα την πολυγλωσσική ικανότητα

του embedding μοντέλου, που αποτελεί κεντρική αρχιτεκτονική επιλογή της εργασίας (Κεφάλαιο 4).

7.2.2 Παρουσίαση του Test Set

Ο Πίνακας 7.1 παρουσιάζει αντιπροσωπευτικό δείγμα του test set. Η πλήρης λίστα των 30 ερωτήσεων διατίθεται στο Παράρτημα Α.

Πίνακας 7.1: Αντιπροσωπευτικό δείγμα του test set

ID	Κατηγορία	Ερώτηση	Στόχος FAQ
D01	direct	Πόσες μέρες έχω για να επιστρέψω ένα προϊόν;	Πολιτική Επιστροφών
P01	paraphrased	Μπορώ να στείλω πίσω κάτι που αγόρασα;	Πολιτική Επιστροφών
P08	paraphrased	Δεν θυμάμαι το password μου	Ξεχασμένος Κωδικός
O04	off-topic	Πώς φτιάχνω σπαγγέτι μπολονέζ;	— (καμία)
X01	cross-lingual	How many days do I have to return a product?	Πολιτική Επιστροφών

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

7.3 Πείραμα 1: Αξιολόγηση του RAG Retrieval

Το πρώτο πείραμα αξιολογεί αποκλειστικά τη συνιστώσα ανάκτησης του συστήματος (RAG), χωρίς εμπλοκή του γλωσσικού μοντέλου. Για κάθε ερώτηση καλείται η συνάρτηση FindRelevantDocs με όριο τα τρία κορυφαία αποτελέσματα, και ελέγχεται αν η αναμενόμενη FAQ εμφανίζεται μεταξύ αυτών. Η μετρική που αξιολογείται είναι το hit rate: το ποσοστό των ερωτήσεων για τις οποίες η σωστή FAQ ανακτήθηκε στις τρεις κορυφαίες θέσεις. Για τις άσχετες ερωτήσεις η αντίστοιχη μετρική είναι το rejection rate: το ποσοστό των ερωτήσεων για τις οποίες το σύστημα σωστά δεν επέστρεψε κανένα σχετικό έγγραφο.

7.3.1 Αποτελέσματα Hit Rate

Πίνακας 7.2: Αποτελέσματα ανάκτησης ανά κατηγορία

Κατηγορία	Ερωτήσεις	Επιτυχίες	Hit Rate
Άμεσες (D01–D10)	10	9	**90%**
Παραφρασμένες (P01–P10)	10	9	**90%**

Κατηγορία	Ερωτήσεις	Επιτυχίες	Hit Rate
Διαγλωσσικές (X01–X05)	5	5	**100%**
Συνολική ακρίβεια (D+P+X)	**25**	**23**	**92%**
Off-topic rejection (O01–O05)	5	0	0%

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

Η συνολική ακρίβεια ανάκτησης 92% (23 σωστές ανακτήσεις στις 25 σχετικές ερωτήσεις) αποτελεί ισχυρό αποτέλεσμα. Ιδιαίτερη σημασία έχει το 100% hit rate στις διαγλωσσικές ερωτήσεις: αγγλικές ερωτήσεις βρήκαν σωστά την αντίστοιχη ελληνική FAQ σε όλες τις πέντε περιπτώσεις, επικυρώνοντας εμπράκτως την αρχιτεκτονική επιλογή του πολυγλωσσικού embedding μοντέλου που τεκμηριώθηκε στο Κεφάλαιο 4.

Χαρακτηριστικό παράδειγμα της πολυγλωσσικής δυνατότητας: η ερώτηση D01 («Πόσες μέρες για επιστροφή;») και η ερώτηση X01 («How many days to return?») έλαβαν αμφότερες ομοιότητα συνημιτόνου 0,916 για την ίδια FAQ «Πολιτική Επιστροφών». Το αποτέλεσμα δείχνει ότι το μοντέλο μεταφέρει τις δύο γλώσσες σχεδόν ταυτόσημα στον ίδιο σημασιολογικό χώρο.

7.3.2 Ανάλυση των Δύο Αποτυχιών Ανάκτησης

Δύο ερωτήσεις απέτυχαν να βρουν τη σωστή FAQ στις τρεις κορυφαίες θέσεις. Η ανάλυση των περιπτώσεων αυτών αναδεικνύει εγγενείς περιορισμούς της σημασιολογικής αναζήτησης μέσω embeddings.

Αποτυχία D10 (Πρόγραμμα Πιστότητας): Η ερώτηση «Πώς λειτουργεί το πρόγραμμα πιστότητας;» επέστρεψε ως κορυφαία αποτελέσματα τις FAQ «Διαθεσιμότητα Προϊόντων» (0,657), «Αξιολογήσεις» (0,639) και «Χρόνος Επιστροφής» (0,635). Η αναμενόμενη FAQ «Πρόγραμμα Πιστότητας ShopEasy Rewards» δεν εμφανίστηκε στις τρεις κορυφαίες θέσεις. Η σύντομη διατύπωση «πρόγραμμα πιστότητας» δεν αντιστοιχήθηκε σημασιολογικά με το πιο εκτενές περιεχόμενο της FAQ, που χρησιμοποιεί εξειδικευμένη ορολογία (πόντοι, επιβράβευση, VIP).

Αποτυχία P04 (Χρόνοι Παράδοσης): Η ερώτηση «Πότε αναμένω να παραλάβω την παραγγελία μου;» επέστρεψε ως κορυφαία αποτελέσματα τις FAQ «Πρόβλημα με Χρέωση» (0,776), «Πολιτική Επιστροφών» (0,760) και «Παρακολούθηση Παραγγελίας» (0,756). Η αναμενόμενη FAQ «Χρόνοι Παράδοσης» δεν εμφανίστηκε. Η φράση «να παραλάβω»

αντιστοιχήθηκε ισχυρότερα με ζητήματα χρέωσης και παρακολούθησης παρά με τους χρόνους παράδοσης καθαυτούς.

Οι δύο αυτές αποτυχίες αποτελούν χαρακτηριστικά παραδείγματα του γνωστού περιορισμού των dense retrieval συστημάτων: όταν η σύντομη διατύπωση ενός ερωτήματος δεν περιέχει επαρκείς σημασιολογικές ενδείξεις, το embedding μοντέλο μπορεί να συνδέσει το ερώτημα με σχετικά αλλά όχι ακριβή έγγραφα. Σε επόμενο πείραμα (ενότητα 7.5) θα δειχθεί ότι ο Human-in-the-Loop μηχανισμός μπορεί να αντισταθμίσει τέτοιες αποτυχίες ανάκτησης χωρίς να απαιτείται επανεκπαίδευση του μοντέλου.

7.3.3 Το Φαινόμενο του 0% Off-topic Rejection

Καμία από τις πέντε άσχετες ερωτήσεις δεν απορρίφθηκε στο επίπεδο ανάκτησης. Για κάθε άσχετη ερώτηση, το σύστημα επέστρεψε τουλάχιστον ένα έγγραφο με ομοιότητα μεγαλύτερη από το όριο 0,5 (MIN_SIMILARITY_SCORE). Οι ομοιότητες όμως ήταν χαμηλές, στο εύρος 0,53–0,65.

Πίνακας 7.3: Συμπεριφορά του RAG σε άσχετες ερωτήσεις

Ερώτηση	**Επιστραφείσα FAQ (top-1)**	**Ομοιότητα**
O01: Πρωθυπουργός	Χρόνοι Παράδοσης	0,649
O02: Πες μια ιστορία	Ξεχασμένος Κωδικός	0,575
O03: Καιρός	Χρόνοι Παράδοσης	0,626
O04: Συνταγή σπαγγέτι	Αυθεντικότητα Προϊόντων	0,557
O05: Ποίημα	Κόστος Αποστολής	0,531

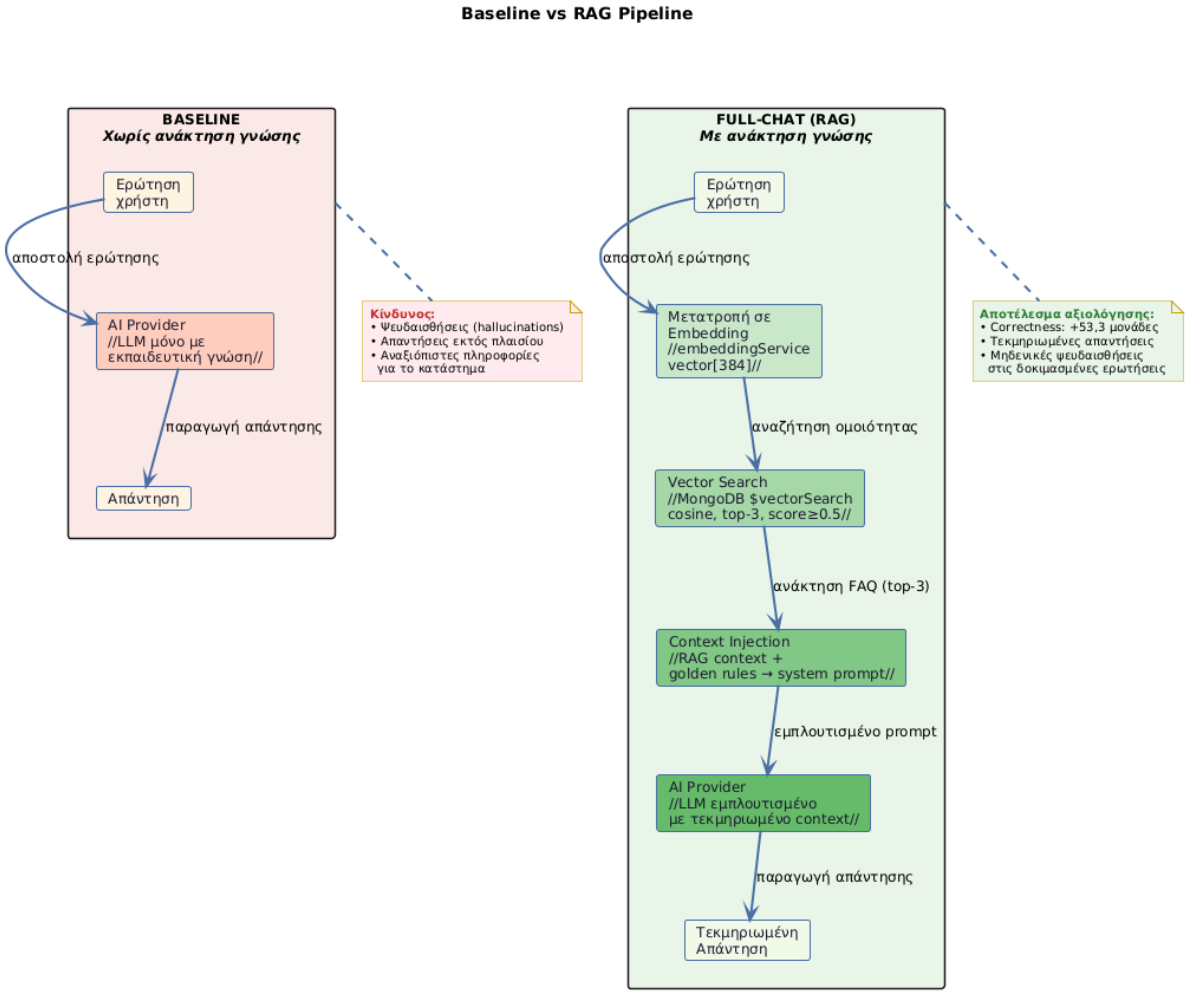
Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

Παρά το αποτέλεσμα αυτό, ο πλήρης βοηθός (Πείραμα 2, ενότητα 7.4) εμφανίζει 100% σωστή απόρριψη των άσχετων ερωτημάτων. Το εύρημα αυτό αναδεικνύει μια σημαντική αρχιτεκτονική ιδιότητα του συστήματος: η απόρριψη άσχετων ερωτημάτων δεν λαμβάνει χώρα στο επίπεδο της ανάκτησης, αλλά στο επίπεδο του γλωσσικού μοντέλου. Ο δυναμικός system prompt δίνει σαφή οδηγία στο μοντέλο ότι ο ρόλος του περιορίζεται στην εξυπηρέτηση πελατών του ShopEasy, και το μοντέλο αναγνωρίζει το άσχετο context ως μη χρήσιμο.

Μια εναλλακτική προσέγγιση θα ήταν η αύξηση του ορίου ομοιότητας (MIN_SIMILARITY_SCORE) σε 0,7. Η επιλογή αυτή θα διόρθωνε την απόρριψη των άσχετων ερωτήσεων, αλλά θα προκαλούσε και απώλεια ορθών παραφρασμένων ανακτήσεων (η P05 βρήκε τη σωστή FAQ με ομοιότητα 0,756, η οποία θα απορριπτόταν με αυστηρότερο

όριο). Επιλέχθηκε λοιπόν η διατήρηση του ορίου στο 0,5 και η εξάρτηση από το γλωσσικό μοντέλο για την αναγνώριση άσχετου context: μια αρχιτεκτονική επιλογή πολλαπλών επιπέδων ασφάλειας.

7.4 Πείραμα 2: Σύγκριση Full System έναντι Baseline



Διάγραμμα 7.1: Baseline vs RAG Pipeline

Το δεύτερο πείραμα αποτελεί ablation study που μετρά τη συνολική συνεισφορά του RAG στην ακρίβεια του chatbot. Δύο εκδοχές του συστήματος εκτελέστηκαν στις ίδιες 30 ερωτήσεις, χρησιμοποιώντας το ίδιο γλωσσικό μοντέλο (LLaMA 3.3-70b μέσω Groq) και τις ίδιες παραμέτρους.

Full-chat: Κάθε ερώτηση περνά μέσα από το πλήρες RAG pipeline. Οι τρεις κορυφαίες FAQ εγχέονται στο dynamic system prompt και το γλωσσικό μοντέλο παράγει την απάντηση έχοντας πρόσβαση στη γνωσιακή βάση.

Baseline: Κάθε ερώτηση στέλνεται απευθείας στο γλωσσικό μοντέλο με ελάχιστο system prompt («Είσαι assistant για το ηλεκτρονικό κατάστημα ShopEasy»). Καμία πρόσβαση στη γνωσιακή βάση, καμία έγχυση context, καμία διόρθωση από feedback.

7.4.1 Κριτήρια Βαθμολόγησης

Οι 60 απαντήσεις του συστήματος (30 από κάθε εκδοχή) εξήχθησαν σε αρχεία CSV και βαθμολογήθηκαν χειροκίνητα. Μια απάντηση κρίθηκε ως σωστή (βαθμός 1) αν περιείχε ακριβείς πληροφορίες σύμφωνες με τη γνωσιακή βάση (σωστούς αριθμούς, σωστές διαδικασίες) και για τις άσχετες ερωτήσεις αν το σύστημα αρνιόταν ευγενικά να απαντήσει. Ως λανθασμένη (βαθμός 0) θεωρήθηκε κάθε απάντηση που περιείχε εσφαλμένα στοιχεία, εφευρεμένες πληροφορίες (ψευδαισθήσεις), ή απαντούσε σε εκτός θέματος ερώτηση.

7.4.2 Συνολικά Αποτελέσματα

Πίνακας 7.4: Συνολική ακρίβεια full-chat έναντι baseline

Σύστημα	**Σωστές απαντήσεις**	**Ακρίβεια**
Full-chat (RAG + AI)	28 / 30	**93,3%**
Baseline (μόνο AI)	12 / 30	**40,0%**
Βελτίωση λόγω RAG	**+16 απαντήσεις**	**+53,3 ποσοστιαίες μονάδες**

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

Η διαφορά 53,3 ποσοστιαίων μονάδων μεταξύ των δύο εκδοχών αποτελεί ισχυρή εμπειρική απόδειξη της προστιθέμενης αξίας του RAG. Το γλωσσικό μοντέλο από μόνο του απάντησε σωστά μόλις στο 40% των ερωτήσεων, ενώ με τη συνδρομή της γνωσιακής βάσης η ακρίβεια εκτοξεύτηκε στο 93,3%.

7.4.3 Ανάλυση ανά Κατηγορία

Η ανάλυση ανά κατηγορία ερωτήσεων (Πίνακας 7.5) δείχνει ότι η βελτίωση είναι ομοιόμορφη σε όλους τους τύπους ερωτημάτων, με ιδιαίτερα μεγάλη διαφορά στις άσχετες και τις διαγλωσσικές ερωτήσεις.

Πίνακας 7.5: Ακρίβεια ανά κατηγορία ερώτησης

Κατηγορία	Full-chat	Baseline	Βελτίωση
Direct (10 ερωτ.)	90%	50%	+40 πμ
Paraphrased (10 ερωτ.)	90%	50%	+40 πμ
Off-topic (5 ερωτ.)	100%	20%	+80 πμ
Cross-lingual (5 ερωτ.)	100%	20%	+80 πμ

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

7.4.4 Τυπικές Ψευδαισθήσεις του Baseline

Η ποσοτική διαφορά 53,3 ποσοστιαίων μονάδων αποκτά πραγματικό νόημα όταν εξετάζονται τα συγκεκριμένα είδη σφαλμάτων που εμφάνισε το baseline. Οι ψευδαισθήσεις δεν ήταν τυχαίες ή απλές παραλήψεις, αλλά συστηματικές και πολύ συγκεκριμένες:

- Λανθασμένοι αριθμοί: Η ερώτηση για το χρονικό περιθώριο επιστροφής (30 ημέρες) απαντήθηκε με «14 ημέρες». Το όριο δωρεάν αποστολής (39€) απαντήθηκε ως «50€» με κόστος μεταφορικών «5€» αντί 3,50€. Η προθεσμία για ελαττωματικά προϊόντα (48 ώρες) απαντήθηκε ως «3 ημέρες».
- Λανθασμένο νόμισμα: Στην αγγλική ερώτηση X02 για το κόστος αποστολής, το baseline απάντησε «$50 threshold, $7.99 shipping» χρησιμοποιώντας δολάρια αντί ευρώ, σαν να επρόκειτο για αμερικανικό κατάστημα.
- Εφευρεμένες μεθόδους πληρωμής: Στην αγγλική ερώτηση X04 για τρόπους πληρωμής, το baseline απάντησε «Net Banking, UPI (Google Pay, PhonePe, BHIM)». Πρόκειται για ινδικά συστήματα πληρωμών, εντελώς άσχετα με το ShopEasy ή την ελληνική αγορά. Το φαινόμενο αποδίδεται στο γεγονός ότι το LLaMA έχει εκπαιδευτεί σε δεδομένα που περιλαμβάνουν εκτενώς το ινδικό e-commerce.
- Εφευρεμένες URL και διαδικασίες: Το baseline επινόησε το URL «www.shopeasy.gr» (το οποίο δεν υπάρχει) και πρόσφερε τραπεζική μεταφορά ως τρόπο πληρωμής, παρότι η σχετική FAQ ρητά αναφέρει ότι οι τραπεζικές μεταφορές δεν γίνονται δεκτές για λιανικές αγορές.
- Απαντήσεις σε άσχετες ερωτήσεις: Το baseline απάντησε στην ερώτηση για τον πρωθυπουργό της Ελλάδας («Κυριάκος Μητσοτάκης»), έγραψε ολόκληρη ιστορία 500 λέξεων όταν του ζητήθηκε, και έδωσε πλήρη συνταγή για σπαγγέτι μπολονέζ. Καμία από αυτές δεν αποτελεί κατάλληλη απάντηση για ένα σύστημα εξυπηρέτησης πελατών e-shop.

Οι παραπάνω ψευδαισθήσεις δεν αποτελούν ακραίες περιπτώσεις: είναι ακριβώς το είδος σφαλμάτων που θα κατέστρεφαν την αξιοπιστία ενός πραγματικού συστήματος εξυπηρέτησης πελατών. Ένας πελάτης που λάβει λανθασμένη πληροφορία για την πολιτική επιστροφών θα χάσει την εμπιστοσύνη του προς το κατάστημα, ενώ η εμφάνιση ινδικών μεθόδων πληρωμής σε ένα ελληνικό e-shop θα μπορούσε να εκληφθεί ως ένδειξη απάτης. Η αρχιτεκτονική RAG,

παρέχοντας πραγματικό context στο γλωσσικό μοντέλο, αποκλείει συστηματικά αυτές τις ψευδαισθήσεις.

7.4.5 Οι Δύο Αποτυχίες του Full-chat

Οι δύο μοναδικές λανθασμένες απαντήσεις του full-chat (D10 και P04) αντιστοιχούν ακριβώς στις δύο αποτυχίες ανάκτησης του Πειράματος 1. Όταν η ανάκτηση επιστρέφει λάθος FAQ, το γλωσσικό μοντέλο είτε δίνει λανθασμένη απάντηση βασισμένη στο εσφαλμένο context, είτε καταφεύγει στην ασφαλή απάντηση «θα σας συνδέσω με την υποστήριξη». Καμιά από τις δύο δεν είναι ψευδαίσθηση: είναι αποτυχία ορθής ανάκτησης. Στο Πείραμα 3 (ενότητα 7.5) δείχνεται πώς ο μηχανισμός HitL μπορεί να αντισταθμίσει αυτές τις περιπτώσεις.

7.5 Πείραμα 3: Επικύρωση του Feedback Loop

Το τρίτο πείραμα αξιολογεί τον πυρήνα του Human-in-the-Loop μηχανισμού: την ικανότητα του συστήματος να ενσωματώνει νέες πληροφορίες ή διορθώσεις σε πραγματικό χρόνο, χωρίς επανεκπαίδευση ή τροποποίηση κώδικα. Το πείραμα προσομοιώνει ένα ρεαλιστικό σενάριο στο οποίο μια επιχείρηση αλλάζει πέντε επιχειρησιακές πολιτικές και πρέπει το chatbot να ενημερωθεί άμεσα.

7.5.1 Μεθοδολογία του Πειράματος

Το πείραμα οργανώνεται σε τρεις φάσεις. Στη φάση Before, οι πέντε ερωτήσεις-σενάρια υποβάλλονται στο σύστημα και καταγράφονται οι απαντήσεις. Στη φάση Inject Corrections, εισάγονται στη MongoDB πέντε εγκεκριμένες διορθώσεις (status: "approved") που αντιστοιχούν στις νέες επιχειρησιακές πολιτικές. Στη φάση After, οι ίδιες πέντε ερωτήσεις υποβάλλονται ξανά, αλλά με νέα session IDs ώστε να μην επηρεάσει το ιστορικό συνομιλίας τα αποτελέσματα.

Η αυτόματη εισαγωγή των διορθώσεων με status «approved» προσομοιώνει τη δράση του διαχειριστή στο admin panel χωρίς διέλευση από την HTTP διεπαφή. Αυτή είναι θεμιτή απλοποίηση για το πείραμα: το ίδιο αποτέλεσμα θα προέκυπτε αν ο διαχειριστής πατούσε «Approve» σε ένα pending feedback.

7.5.2 Τα Πέντε Σενάρια Δοκιμής

Επιλέχθηκαν πέντε ρεαλιστικές αλλαγές πολιτικής που αντικατοπτρίζουν συνηθισμένες ενημερώσεις σε ένα e-shop:

Πίνακας 7.6: Σενάρια του πειράματος feedback loop

#	Αλλαγή Πολιτικής	Νέα Πληροφορία προς Έγχυση
1	Επιστροφές απαιτούν πλέον φόρμα ΕΠ-2025	«φόρμα ΕΠ-2025», συμπληρώνεται πριν την αποστολή
2	Όριο δωρεάν αποστολής: 39€ → 25€	Νέο όριο 25€, παλιό 39€ μη ισχύον
3	Ελαττωματικά: κλήση στο 210-9000000 για αριθμό υπόθεσης	Τηλέφωνο «210-9000000», αναφορά εντός 24 ωρών
4	Κατώφλι VIP πιστότητας: 500 → 300 πόντοι	«300 πόντοι», 15% έκπτωση
5	Αποδοχή Bitcoin/Ethereum μέσω CryptoPay	«Bitcoin», «Ethereum», «CryptoPay»

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

7.5.3 Αποτελέσματα

Σε όλα τα πέντε σενάρια, οι ενημερωμένες απαντήσεις του chatbot αντικατόπτριζαν πλήρως τις εγκεκριμένες διορθώσεις. Ποσοστό υιοθέτησης: 5/5 (100%).

Πίνακας 7.7: Σύγκριση απαντήσεων πριν και μετά την έγχυση διορθώσεων

#	Πριν	Μετά	Υιοθέτηση
1	Ανέφερε 30 ημέρες, καμία φόρμα	Ρητή αναφορά στη φόρμα ΕΠ-2025 ως υποχρεωτική	**Ναι**
2	«Δωρεάν για παραγγελίες άνω των 39€»	«Δωρεάν για παραγγελίες άνω των 25€ (αλλαγή από 39€)»	**Ναι**
3	«Επικοινωνία εντός 48 ωρών»	«Κλήση στο 210-9000000, αναφορά εντός 24 ωρών»	**Ναι**
4	Παραπομπή στην υποστήριξη (αποτυχία ανάκτησης D10)	«VIP status στους 300 πόντους με 15% έκπτωση»	**Ναι**
5	Παρέθεσε τυπικές μεθόδους, χωρίς κρυπτονομίσματα	«Bitcoin και Ethereum γίνονται δεκτά μέσω CryptoPay»	**Ναι**

Πηγή: Ιδία επεξεργασία (πειραματικά αποτελέσματα).

7.5.4 Το Σενάριο 4: Ο Feedback Loop ως Δίχτυ Ασφαλείας

Το Σενάριο 4 παρουσιάζει ιδιαίτερο ενδιαφέρον και αξίζει ξεχωριστής συζήτησης. Στη φάση Before, η ερώτηση σχετικά με το πρόγραμμα πιστότητας δεν απαντήθηκε σωστά: όπως φάνηκε στο Πείραμα 1 (αποτυχία D10), η σχετική FAQ δεν ανακτήθηκε από το RAG, και το σύστημα κατέφυγε σε γενική απάντηση παραπομπής στην υποστήριξη. Στη φάση After, μετά την έγχυση της εγκεκριμένης διόρθωσης που περιείχε τις σωστές πληροφορίες του προγράμματος VIP, το σύστημα απάντησε με ακρίβεια.

Το εύρημα αυτό αναδεικνύει μια μη προφανή αρχιτεκτονική ιδιότητα του συστήματος: ο μηχανισμός HitL δεν είναι μόνο εργαλείο διόρθωσης λανθασμένων απαντήσεων, αλλά λειτουργεί επίσης ως δίχτυ ασφαλείας για αποτυχίες ανάκτησης. Όταν το RAG δεν βρίσκει τη σχετική FAQ, μια εγκεκριμένη ανθρώπινη διόρθωση μπορεί να προσφέρει την πληροφορία απευθείας μέσω του dynamic system prompt. Έτσι, τα δύο κύρια τεχνικά στοιχεία της εργασίας (RAG και HitL) δεν είναι ανεξάρτητα, αλλά συνεργαζόμενα: το ένα αντισταθμίζει τις αδυναμίες του άλλου.

7.5.5 Τεχνική Λειτουργία της Έγχυσης

Ο μηχανισμός που επιτρέπει την άμεση υιοθέτηση των διορθώσεων βασίζεται σε δύο συνιστώσες. Αρχικά, ο feedbackEngine ανακτά τις εγκεκριμένες διορθώσεις από τη MongoDB (όριο 5, ταξινόμηση από νεότερα προς παλαιότερα) σε κάθε αίτημα chat. Στη συνέχεια, οι διορθώσεις αυτές μορφοποιούνται ως few-shot παραδείγματα και προστίθενται στο system prompt με τη μορφή:

```
ΣΗΜΑΝΤΙΚΕΣ ΔΙΟΡΘΩΣΕΙΣ (εγκεκριμένες από διαχειριστή): - Ε: «Πόσες μέρες για επιστροφή;» Α: «...φόρμα ΕΠ-2025 απαιτείται...» - Ε: «Δωρεάν αποστολή;» Α: «...όριο 25€...»
```

Το γλωσσικό μοντέλο διαβάζει αυτές τις διορθώσεις ως μέρος του context του και τις θεωρεί ως υψηλής προτεραιότητας οδηγία. Δεν απαιτείται επανεκπαίδευση, fine-tuning ή τροποποίηση κώδικα: μόλις ο διαχειριστής εγκρίνει μια διόρθωση από το admin panel, η νέα πληροφορία υπάρχει στις απαντήσεις της επόμενης χρονικής στιγμής για όλους τους χρήστες, σε όλα τα sessions. Πρόκειται για ταχύτερη και οικονομικότερη μέθοδο ενημέρωσης γνώσης από οποιαδήποτε εναλλακτική που θα απαιτούσε εκπαίδευση μοντέλου.

7.6 Συζήτηση Ευρημάτων και Περιορισμοί

Τα τρία πειράματα παρέχουν συμπληρωματική εικόνα της απόδοσης του συστήματος. Στο επίπεδο της ανάκτησης, η σημασιολογική αναζήτηση επιτυγχάνει 92% ακρίβεια σε σχετικές ερωτήσεις, με ιδιαίτερα ισχυρή απόδοση στη διαγλωσσική αναζήτηση (100%). Στο επίπεδο του πλήρους συστήματος, το RAG προσφέρει βελτίωση 53,3 ποσοστιαίων μονάδων έναντι του baseline, με τους τύπους ψευδαισθήσεων που αποτρέπει να είναι ακριβώς εκείνοι που θα κατέστρεφαν την αξιοπιστία ενός πραγματικού συστήματος (λανθασμένες πολιτικές, εφευρεμένες URL, λάθος νόμισμα). Στο επίπεδο του feedback loop, και τα πέντε σενάρια αλλαγής πολιτικής υιοθετήθηκαν επιτυχώς, με μία μάλιστα περίπτωση όπου ο HitL αντιστάθμισε αποτυχία της ανάκτησης.

7.6.1 Περιορισμοί της Αξιολόγησης

Η αξιολόγηση που παρουσιάστηκε εμπεριέχει αρκετούς περιορισμούς που πρέπει να αναγνωριστούν για να ερμηνευθούν σωστά τα αποτελέσματα.

Μέγεθος δείγματος. Το test set των 30 ερωτήσεων είναι αρκετά μεγάλο για να δώσει ενδείξεις τάσης, αλλά μικρό για στατιστικά σημαντικά συμπεράσματα. Διαστήματα εμπιστοσύνης και έλεγχοι σημαντικότητας (π.χ. McNemar test για paired comparison) θα απαιτούσαν μεγαλύτερο δείγμα. Τα παρουσιαζόμενα ποσοστά πρέπει να ερμηνεύονται ως ενδείξεις απόδοσης και όχι ως οριστικά μετρήσεις.

Μονός αξιολογητής. Η χειροκίνητη βαθμολόγηση των απαντήσεων έγινε από έναν αξιολογητή (τον συγγραφέα). Η ακαδημαϊκή πρακτική σε αξιολογήσεις NLP συστημάτων προτείνει τουλάχιστον δύο ανεξάρτητους αξιολογητές με υπολογισμό μέτρου συμφωνίας (Cohen's kappa). Η μονο-αξιολόγηση εισάγει υποκειμενικότητα στα οριακά παραδείγματα.

Πεδίο δοκιμής. Όλα τα πειράματα εκτελέστηκαν με τη γνωσιακή βάση των 25 FAQ του ShopEasy. Η συμπεριφορά του συστήματος σε σημαντικά μεγαλύτερες βάσεις (εκατοντάδες ή χιλιάδες FAQ) δεν δοκιμάστηκε, και τα αποτελέσματα δεν μπορούν να γενικευτούν χωρίς πρόσθετα πειράματα.

Επιλογή μοντέλου. Τα πειράματα εκτελέστηκαν με το LLaMA 3.3-70b. Άλλα γλωσσικά μοντέλα (π.χ. GPT-4, Gemini 2.0 Pro) ενδέχεται να δίνουν διαφορετικά αποτελέσματα τόσο στο baseline (πιθανώς υψηλότερη ακρίβεια λόγω εκτενέστερης εκπαίδευσης) όσο και στο full-chat (πιθανώς καλύτερη ή χειρότερη χρήση του παρεχόμενου context).

Πειραματικός σχεδιασμός του feedback loop. Στο Πείραμα 3, η αναμενόμενη απάντηση και η εγκεκριμένη διόρθωση γράφτηκαν από τον συγγραφέα. Πρόκειται για ελεγχόμενες συνθήκες που δείχνουν τον μηχανισμό υιοθέτησης διορθώσεων, αλλά δεν αξιολογούν πλήρως τη γενίκευση: μια πιο αυστηρή δοκιμή θα έβαζε διόρθωση σε μία διατύπωση και θα έλεγχε αν εφαρμόζεται σε διαφορετικά παραφρασμένα ερωτήματα. Το Σενάριο 4 (D10) προσεγγίζει αυτό το είδος δοκιμής, καθώς η διόρθωση εφαρμόστηκε σε ερώτηση που είχε αρχικά αποτύχει η ανάκτηση.

7.6.2 Αρχιτεκτονικά Συμπεράσματα

Πέρα από τα αριθμητικά αποτελέσματα, η αξιολόγηση οδήγησε σε τρία αρχιτεκτονικά συμπεράσματα που έχουν αξία για παρόμοιες υλοποιήσεις:

Ο διαχωρισμός μεταξύ ανάκτησης και απόρριψης άσχετων ερωτημάτων είναι σημαντικός. Η ανάκτηση τείνει να επιστρέφει πάντοτε κάτι, ακόμα και για άσχετες ερωτήσεις. Η αρχιτεκτονική ασφάλειας πρέπει να βασίζεται στο γλωσσικό μοντέλο και τον system prompt, όχι στο όριο ομοιότητας μόνο. Η ρύθμιση του ορίου σε σχετικά χαλαρό επίπεδο (0,5) ευνοεί την ανάκληση παραφρασμένων ερωτημάτων εις βάρος της απόρριψης άσχετων.

Τα δύο κύρια τεχνικά στοιχεία της εργασίας (RAG και HitL) λειτουργούν συμπληρωματικά μεταξύ τους. Όταν η ανάκτηση αποτυγχάνει (όπως στο D10), ο HitL μπορεί να αντισταθμίσει με εγκεκριμένη διόρθωση. Αντίστροφα, ο HitL απαιτεί ένα λειτουργικό RAG για να εστιάσει σε πραγματικές αποτυχίες αντί να διορθώνει συστηματικά λάθη του συστήματος.

Το κόστος του inference-time alignment είναι σημαντικά χαμηλότερο από εναλλακτικές προσεγγίσεις όπως το fine-tuning. Η ενσωμάτωση πέντε νέων πολιτικών στο σύστημα έγινε με αποθήκευση πέντε εγγραφών στη MongoDB. Καμία εκπαίδευση μοντέλου, καμία ενημέρωση κώδικα, και η αλλαγή ισχύει άμεσα για όλους τους χρήστες.

7.7 Σύνδεση με τα Ερευνητικά Ερωτήματα

Τα ευρήματα της αξιολόγησης απαντούν συγκεκριμένα στα τρία ερευνητικά ερωτήματα που τέθηκαν στην Εισαγωγή.

RQ1: Αποτελεσματικότητα του RAG έναντι των ψευδαισθήσεων. Τα Πειράματα 1 και 2 παρέχουν άμεση εμπειρική απάντηση. Η ανάκτηση πετυχαίνει 92% ακρίβεια σε σχετικές ερωτήσεις και 100% ακρίβεια στις διαγλωσσικές, επιβεβαιώνοντας τη διαγλωσσική ικανότητα του μοντέλου (Reimers & Gurevych, 2020), ενώ η συνολική απόδοση toufull-chat (93,3%) είναι πάνω από διπλάσια από το baseline (40,0%). Οι ψευδαισθήσεις που αποτράπηκαν είναι συστηματικές και κρίσιμες (λανθασμένες πολιτικές, εφευρεμένες URL, λάθος νόμισμα, ινδικά συστήματα πληρωμών σε ελληνικό κατάστημα). Το RQ1 απαντάται καταφατικά: το RAG είναι αποτελεσματικός μηχανισμός μείωσης ψευδαισθήσεων σε αυτό το πεδίο εφαρμογής.

RQ2: Inference-time alignment μέσω HitL. Το Πείραμα 3 παρέχει την απάντηση. Πέντε διαφορετικές αλλαγές πολιτικής υιοθετήθηκαν επιτυχώς από το σύστημα μέσα από εγκεκριμένες διορθώσεις, χωρίς επανεκπαίδευση και χωρίς τροποποίηση κώδικα. Το Σενάριο 4 έδειξε επιπλέον ότι ο μηχανισμός μπορεί να αντισταθμίσει αποτυχίες της ανάκτησης. Το RQ2 απαντάται καταφατικά: ο HitL ως μηχανισμός inference-time alignment λειτουργεί στην πράξη και προσφέρει σημαντική προστιθέμενη αξία.

RQ3: Βιωσιμότητα ανοιχτής αρχιτεκτονικής. Η απάντηση εδώ συντίθεται από όλα τα προηγούμενα ευρήματα. Με 93,3% συνολική ακρίβεια, λειτουργικό μηχανισμό συνεχούς βελτίωσης, και συνολικό κόστος εκτέλεσης στα όρια των δωρεάν επιπέδων των υπηρεσιών (Groq, MongoDB Atlas Free Tier), η αρχιτεκτονική αποδεικνύεται τεχνικά εφικτή για μικρές και μεσαίες επιχειρήσεις. Το RQ3 απαντάται καταφατικά υπό προϋποθέσεις: η ανοιχτή αρχιτεκτονική είναι βιώσιμη για περιορισμένη κλίμακα και απλούστερα use cases, ενώ μεγαλύτερα περιβάλλοντα παραγωγής θα απαιτούσαν πρόσθετη μηχανική (κλιμάκωση, παρακολούθηση, ασφάλεια) που δεν αξιολογήθηκε στην παρούσα εργασία.

7.8 Σύνοψη Κεφαλαίου

Το παρόν κεφάλαιο παρουσίασε την ποσοτική αξιολόγηση του ShopEasy ChatBot μέσα από τρία συμπληρωματικά πειράματα. Το πρώτο πείραμα εστίασε στη σημασιολογική ανάκτηση και έδειξε 92% ακρίβεια στις σχετικές ερωτήσεις, με την πολυγλωσσική αναζήτηση να επιτυγχάνει 100% hit rate. Το δεύτερο πείραμα συνέκρινε το πλήρες σύστημα με baseline χωρίς RAG και κατέγραψε βελτίωση 53,3 ποσοστιαίων μονάδων, με συγκεκριμένα παραδείγματα ψευδαισθήσεων που το RAG αποτρέπει. Το τρίτο πείραμα επικύρωσε τον μηχανισμό HitL με 100% υιοθέτηση πέντε διαφορετικών διορθώσεων.

Η πιο αξιοσημείωτη ανακάλυψη της αξιολόγησης αφορά τη σχέση μεταξύ των δύο κύριων τεχνικών στοιχείων, παρά κάποια μεμονωμένη μετρική: το RAG και ο HitL λειτουργούν ως αλληλένδετα συστήματα, με τον HitL να μπορεί να αντισταθμίσει αποτυχίες της ανάκτησης (παράδειγμα: σενάριο 4). Πέρα από τα ποσοτικά αποτελέσματα, αναγνωρίστηκαν αρκετοί περιορισμοί της αξιολόγησης, με κυριότερους το μικρό μέγεθος δείγματος και τη χρήση μονού αξιολογητή. Παρά τους περιορισμούς αυτούς, τα ευρήματα παρέχουν επαρκή εμπειρική απάντηση στα τρία ερευνητικά ερωτήματα της εργασίας.

8. Συμπεράσματα και Μελλοντική Δουλειά

8.1 Σύνοψη της Εργασίας

Η παρούσα εργασία είχε ως στόχο τη σχεδίαση και υλοποίηση ενός AI chatbot εξυπηρέτησης πελατών για το φανταστικό ηλεκτρονικό κατάστημα ShopEasy, με έμφαση σε δύο τεχνικές: Retrieval-Augmented Generation (RAG) για ανάκτηση σχετικής πληροφορίας από γνωσιακή βάση, και Human-in-the-Loop (HitL) feedback loop για βελτίωση του συστήματος χωρίς επανεκπαίδευση.

Το αποτέλεσμα είναι ένα λειτουργικό σύστημα τριών επιπέδων (React frontend, Node.js/Express backend, MongoDB Atlas) που τρέχει τοπικά μέσω Docker. Οι κεντρικές τεχνικές επιλογές (τοπικό μοντέλο embeddings, factory pattern για εναλλαξιμότητα παρόχων AI, δυναμικό system prompt) αποδείχθηκαν υλοποιήσιμες στα πλαίσια μιας προπτυχιακής εργασίας, αν και με αρκετές δυσκολίες που τεκμηριώνονται στο Κεφ. 6.10.

8.2 Τι Λειτούργησε και τι Όχι

8.2.1 Θετικά Αποτελέσματα

RAG με τοπικά embeddings. Η χρήση του paraphrase-multilingual-MiniLM-L12-v2 μέσω @xenova/transformers απέδωσε ικανοποιητική σημασιολογική αναζήτηση χωρίς cloud API. Το μοντέλο χειρίζεται cross-lingual queries (ερωτήσεις στα αγγλικά βρίσκουν εγγραφές στα ελληνικά), κάτι που ήταν απρόσμενο θετικό αποτέλεσμα κατά τις δοκιμές.

HitL feedback loop. Ο κύκλος αξιολόγηση → διόρθωση → έγκριση → injection λειτουργεί στην πράξη. Μετά από μερικές εγκεκριμένες διορθώσεις, το σύστημα άρχισε να αντιμετωπίζει σωστά ερωτήματα που αρχικά απαντούσε λανθασμένα.

Factory pattern για παρόχους AI. Η αφαίρεση του παρόχου μέσω του AIProvider interface λειτούργησε καλά: η εναλλαγή μεταξύ OpenRouter, Gemini και OpenAI γίνεται αποκλειστικά μέσω περιβαλλοντικής μεταβλητής, χωρίς αλλαγή κώδικα.

8.2.2 Περιορισμοί και Αδυναμίες

Μη ποσοτική αξιολόγηση. Η εργασία δεν περιλαμβάνει structured αξιολόγηση με μετρικές (precision, recall, BLEU score κτλ.). Τα συμπεράσματα για την αποτελεσματικότητα βασίζονται σε ποιοτικές παρατηρήσεις κατά τις δοκιμές, κάτι που αποτελεί βασικό περιορισμό.

Προσομοίωση RLHF, όχι πραγματικό RL. Ο HitL μηχανισμός προσομοιώνει το Reward Modeling στάδιο του RLHF μέσω few-shot injection, αλλά δεν υλοποιεί πραγματικό reinforcement learning. Δεν υπάρχει reward function, δεν ενημερώνονται τα βάρη του μοντέλου, και η "μάθηση" χάνεται αν διαγραφεί η MongoDB collection.

Μόνο τοπική λειτουργία. Το σύστημα δεν έχει δοκιμαστεί σε παραγωγικό περιβάλλον με πραγματικούς χρήστες. Θέματα κλιμάκωσης, ασφάλειας και ταυτόχρονης εξυπηρέτησης πολλών χρηστών παραμένουν αναντιμέτωπα.

Fallback αναζήτηση. Σε τοπικό MongoDB (εκτός Atlas), το σύστημα υποβαθμίζεται σε keyword search που χάνει σημαντικό μέρος της σημασιολογικής λειτουργικότητας.

8.3 Συνεισφορά της Εργασίας

Η κύρια συνεισφορά είναι πρακτική, όχι θεωρητική: η εργασία δείχνει ότι μια αρχιτεκτονική RAG + HitL μπορεί να υλοποιηθεί πλήρως με open-source εργαλεία, τοπικά, σε εύλογο χρόνο ανάπτυξης. Η εργασία τεκμηριώνει πρακτικά εμπόδια (Docker caching, embedding mismatch, prompt overflow) που σπάνια αναφέρονται σε ακαδημαϊκό πλαίσιο.

Επιπλέον, ο συνδυασμός MongoDB Atlas ως ενιαίας βάσης για έγγραφα και διανύσματα μειώνει την αρχιτεκτονική πολυπλοκότητα σε σχέση με λύσεις που χρησιμοποιούν ξεχωριστή vector database (π.χ. Pinecone, Weaviate), κάτι που φαίνεται να είναι χρήσιμο trade-off για μικρής κλίμακας εφαρμογές.

8.4 Μελλοντική Δουλειά

Ποσοτική αξιολόγηση. Σχεδιασμός structured test set με γνωστές σωστές απαντήσεις και μέτρηση precision/recall του RAG pipeline πριν και μετά το feedback loop.

Αυτόματη επανεκπαίδευση embeddings. Χρήση των εγκεκριμένων feedback ως training data για fine-tuning του embedding model, κλείνοντας τον κύκλο μάθησης πιο αυτόματα.

Agentic δυνατότητες. Επέκταση με tool use (function calling) για πραγματικές ενέργειες: αναζήτηση παραγγελίας, αποστολή email επιβεβαίωσης, δημιουργία ticket.

Παραγωγική ανάπτυξη. Προσθήκη authentication, rate limiting, logging και deployment σε cloud περιβάλλον για πραγματική χρήση.

Πολυτροπική υποστήριξη. Ενσωμάτωση εικόνων στο knowledge base και αναζήτηση μέσω multimodal embeddings.

8.5 Κλείσιμο

Συνολικά, η εργασία πέτυχε τον πρωταρχικό στόχο της: να δείξει ότι ένα σύστημα RAG με HitL feedback μπορεί να υλοποιηθεί από μηδέν με διαθέσιμα open-source εργαλεία. Τα τεχνικά εμπόδια που αντιμετωπίστηκαν υποδεικνύουν ότι το μεγαλύτερο κόστος ανάπτυξης τέτοιων συστημάτων εντοπίζεται στη διαχείριση της υποδομής και όχι στην αλγοριθμική πολυπλοκότητα.

9. Βιβλιογραφία

Ελληνόγλωσση βιβλιογραφία

Δεν χρησιμοποιήθηκαν ελληνόγλωσσες ακαδημαϊκές πηγές.

Ξενόγλωσση βιβλιογραφία

Castillo, D., & Farrugia Caruana, L. (2025). Unveiling customer expectations of chatbot interactions: A systematic literature review and research agenda. *International Journal of Human–Computer Interaction*. Advance online publication. https://doi.org/10.1080/10447318.2025.2588815

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 (NAACL-HLT 2019)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). *Retrieval-augmented generation for large language models: A survey* [Preprint]. arXiv. https://arxiv.org/abs/2312.10997

Gartner. (2022, July 27). *Gartner predicts chatbots will become a primary customer service channel within five years* [Press release]. https://www.gartner.com/en/newsroom/press-releases/2022-07-27-gartner-predicts-chatbots-will-become-a-primary-customer-service-channel-within-five-years

Grand View Research. (2025). *Conversational AI market size, share & trends analysis report, 2025–2030*. https://www.grandviewresearch.com/industry-analysis/conversational-ai-market-report

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models* [Preprint]. arXiv. https://arxiv.org/abs/2001.08361

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33* (pp. 9459–9474). https://arxiv.org/abs/2005.11401

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473 (Preprint: https://arxiv.org/abs/1603.09320)

Marcineková, K., Sujová, A. J., & Ďurica, R. (2025). Implementing AI chatbots in customer service optimization—A case study in micro-enterprise. *Information, 16*(12), Article 1078. https://doi.org/10.3390/info16121078

McKinsey & Company. (2022). *The next frontier of customer engagement: AI-enabled customer service*. https://www.mckinsey.com/capabilities/operations/our-insights/the-next-frontier-of-customer-engagement-ai-enabled-customer-service

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space* [Preprint]. arXiv. https://arxiv.org/abs/1301.3781

MongoDB. (2024). *Retrieval-augmented generation (RAG) with Atlas Vector Search*. MongoDB Documentation. https://www.mongodb.com/docs/atlas/atlas-vector-search/rag/

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35*. https://arxiv.org/abs/2203.02155

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*. OpenAI. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Reimers, N., & Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 4512–4525). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.365

Salesforce. (2025). *State of service report* (7th ed.). https://www.salesforce.com/news/stories/state-of-service-report-announcement-2025/

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms* [Preprint]. arXiv. https://arxiv.org/abs/1707.06347

Statista. (2024). *Retail e-commerce sales worldwide from 2014 to 2027*. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

Stanford Institute for Human-Centered AI. (2025). *The 2025 AI Index report*. Stanford University. https://hai.stanford.edu/ai-index/2025-ai-index-report

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30* (pp. 5998–6008). https://arxiv.org/abs/1706.03762

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*. https://arxiv.org/abs/2206.07682

Wiggins, A. (2017). *The twelve-factor app*. https://12factor.net

Zimmerman, J., Forlizzi, J., & Evenson, S. (2007). Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)* (pp. 493–502). Association for Computing Machinery. https://doi.org/10.1145/1240624.1240704

10. Παράρτημα Α: Πλήρες Σύνολο Ελέγχου (30 Ερωτήσεις)

Ο παρακάτω πίνακας παραθέτει το πλήρες σύνολο των 30 ερωτήσεων που χρησιμοποιήθηκαν στην αξιολόγηση (Κεφάλαιο 7), οργανωμένο στις τέσσερις κατηγορίες: Άμεσες (D), Παραφρασμένες (P), Άσχετες (O) και Διαγλωσσικές (X). Για κάθε ερώτηση αναφέρεται η στοχευόμενη εγγραφή της γνωσιακής βάσης (FAQ).

ID	Κατηγορία	Ερώτηση	Γλώσσα	Στόχος FAQ
D01	Άμεση	Πόσες μέρες έχω για να επιστρέψω ένα προϊόν;	EL	Πολιτική Επιστροφών
D02	Άμεση	Πότε θα λάβω τα χρήματά μου πίσω μετά την επιστροφή;	EL	Χρόνος Επιστροφής Χρημάτων
D03	Άμεση	Η αποστολή είναι δωρεάν;	EL	Κόστος Αποστολής
D04	Άμεση	Πόσες μέρες παίρνει η παράδοση;	EL	Χρόνοι Παράδοσης
D05	Άμεση	Πώς μπορώ να παρακολουθήσω την παραγγελία μου;	EL	Παρακολούθηση Παραγγελίας
D06	Άμεση	Ποιες μεθόδους πληρωμής δέχεστε;	EL	Αποδεκτές Μέθοδοι Πληρωμής
D07	Άμεση	Πώς δημιουργώ λογαριασμό;	EL	Δημιουργία Λογαριασμού
D08	Άμεση	Ξέχασα τον κωδικό μου, τι κάνω;	EL	Ξεχασμένος Κωδικός
D09	Άμεση	Τι γίνεται αν παραλάβω ελαττωματικό προϊόν;	EL	Επιστροφή Ελαττωματικού Προϊόντος
D10	Άμεση	Πώς λειτουργεί το πρόγραμμα πιστότητας;	EL	Πρόγραμμα Πιστότητας ShopEasy Rewards
P01	Παραφρασμένη	Μπορώ να στείλω πίσω κάτι που αγόρασα;	EL	Πολιτική Επιστροφών
P02	Παραφρασμένη	Σε πόσες μέρες επιστρέφονται	EL	Χρόνος Επιστροφής Χρημάτων

		τα χρήματα στην κάρτα μου;		
P03	Παραφρασμένη	Υπάρχει δωρεάν μεταφορικά;	EL	Κόστος Αποστολής
P04	Παραφρασμένη	Πότε αναμένω να παραλάβω την παραγγελία μου;	EL	Χρόνοι Παράδοσης
P05	Παραφρασμένη	Έχω αριθμό tracking, πού τον βλέπω;	EL	Παρακολούθηση Παραγγελίας
P06	Παραφρασμένη	Δέχεστε PayPal ή κάρτες;	EL	Αποδεκτές Μέθοδοι Πληρωμής
P07	Παραφρασμένη	Θέλω να φτιάξω νέο λογαριασμό στο ShopEasy	EL	Δημιουργία Λογαριασμού
P08	Παραφρασμένη	Δεν θυμάμαι το password μου	EL	Ξεχασμένος Κωδικός
P09	Παραφρασμένη	Έλαβα κατεστραμμένο προϊόν, τι πρέπει να κάνω;	EL	Επιστροφή Ελαττωματικού Προϊόντος
P10	Παραφρασμένη	Κερδίζω πόντους με κάθε αγορά;	EL	Πρόγραμμα Πιστότητας ShopEasy Rewards
O01	Άσχετη	Ποιος είναι ο πρωθυπουργός της Ελλάδας;	EL	(καμία)
O02	Άσχετη	Πες μου μια ιστορία	EL	(καμία)
O03	Άσχετη	Τι καιρό κάνει σήμερα;	EL	(καμία)
O04	Άσχετη	Πώς φτιάχνω σπαγγέτι μπολονέζ;	EL	(καμία)
O05	Άσχετη	Βοήθησέ με να γράψω ένα ποίημα	EL	(καμία)
X01	Διαγλωσσική	How many days do I have to return a product?	EN	Πολιτική Επιστροφών
X02	Διαγλωσσική	Is shipping free?	EN	Κόστος Αποστολής
X03	Διαγλωσσική	How do I track my order?	EN	Παρακολούθηση Παραγγελίας

X04	Διαγλωσσική	What payment methods do you accept?	EN	Αποδεκτές Μέθοδοι Πληρωμής
X05	Διαγλωσσική	I received a damaged product, what should I do?	EN	Επιστροφή Ελαττωματικού Προϊόντος

Πίνακας Α.1: Πλήρες σύνολο ελέγχου 30 ερωτήσεων ανά κατηγορία

Πηγή: Ιδία επεξεργασία (πειραματικά δεδομένα).