ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Πρόγραμμα Μεταπτυχιακών Σπουδών: Πληροφοριακά & Επικοινωνιακά Συστήματα

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τίτλος : Αυτόματη Παραγωγή UML Διαγραμμάτων Βοηθούμενη από την Τεχνητή Νοημοσύνη

Συντάκτης: Μπουργάνης Βασίλειος
Α.Μ: 352023014

Επιβλέπων Καθηγητής:

Κρητικός Κυριάκος, Αναπληρωτής Καθηγητής

Τριμελής Επιτροπή:

Κ. Κρητικός, Σ. Κοκολάκης, Π. Συμεωνίδης

Σάμος, Ιούλιος 2025

Περιεχόμενα

Λίστα Εικόνων

Λίστα Πινάκων

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την ανάπτυξη μιας διαδικτυακής εφαρμογής που αξιοποιεί τεχνολογίες Τεχνητής Νοημοσύνης και ειδικότερα Μεγάλα Γλωσσικά Μοντέλα (LLMs), για την αυτόματη παραγωγή UML διαγραμμάτων από περιγραφές σε φυσική γλώσσα. Η εφαρμογή επιτρέπει τη δημιουργία τεσσάρων τύπων διαγραμμάτων UML (Class, Sequence, Use Case, Activity), παρέχοντας στους χρήστες τη δυνατότητα να επιλέγουν το επιθυμητό μοντέλο LLM (π.χ. GPT, Gemini, DeepSeek), το μέγεθος του διαγράμματος, καθώς και τη μορφή εξαγωγής του (PNG, SVG, ASCII).

Η λειτουργικότητα της εφαρμογής επεκτείνεται με δυνατότητες επεξεργασίας του παραγόμενου PlantUML κώδικα (δηλαδή του μοντέλου του διαγράμματος), διαχείρισης χρηστών και υποστήριξης πολλαπλών σεναρίων χρήσης. Η αξιολόγηση του συστήματος έγινε με βάση πραγματικά σενάρια, χρησιμοποιώντας μετρικές, όπως η ακρίβεια (precision), η πληρότητα (recall), η συντακτική ορθότητα και ο χρόνος απόκρισης. Τα αποτελέσματα επιβεβαιώνουν την ικανότητα του συστήματος να παράγει διαγράμματα υψηλής ποιότητας, ακόμα και σε σύνθετες περιπτώσεις.

Η εργασία αναδεικνύει τη δυνατότητα ενσωμάτωσης των LLMs σε εργαλεία λογισμικού για την αυτοματοποιημένη μοντελοποίηση και προτείνει προοπτικές μελλοντικής επέκτασής της, όπως υποστήριξη εισόδου από πηγαίο κώδικα, fine-tuning των LLMs σε εξειδικευμένα πεδία, και δημιουργία benchmark dataset για την αξιολόγηση διαφορετικών LLMs ως προς την ικανότητα παραγωγής τους UML διαγραμμάτων.

Λέξεις-κλειδιά:

UML Διαγράμματα, Μεγάλα Γλωσσικά Μοντέλα, Τεχνητή Νοημοσύνη, Επεξεργασία Φυσικής Γλώσσας, Αυτόματη Μοντελοποίηση, PlantUML, Δημιουργία Διαγραμμάτων, Αντίστροφη Μηχανική, Παραγωγή Κώδικα, Στατική Ανάλυση, Μηχανική Προτροπών, Συνεργατική Επεξεργασία, Επεξηγήσιμη Τεχνητή Νοημοσύνη.

Abstract

This thesis focuses on the development of a web application that utilizes Artificial Intelligence technologies—specifically, Large Language Models (LLMs)—to automatically generate UML diagrams from natural language descriptions. The application supports the creation of four UML diagram types (Class, Sequence, Use Case, Activity), offering users the ability to select the desired LLM (e.g., GPT, Gemini, DeepSeek), define the diagram size, and choose the export format (PNG, SVG, ASCII).

The application extends its functionality by enabling user management, feedback provision, support for various usage scenarios, and the direct editing of the generated PlantUML code (i.e., the model of the diagram). System evaluation was conducted using real-world scenarios and metrics, such as precision, recall, syntactic validity, and response time. The evaluation results confirm the system's capability to produce high-quality diagrams, even in complex cases.

This work highlights the potential for integrating LLMs into software tools for automated modeling and suggests directions for its future extension, such as code-to-diagram conversion, domain-specific fine-tuning of LLMs, and the creation of benchmark datasets for evaluating the UML diagram generation capabilities of various LLMs.

Keywords:

UML Diagrams, Large Language Models, Artificial Intelligence, Natural Language Processing, Automatic Modelling, PlantUML, Diagram Generation, Reverse Engineering, Code Generation, Static Analysis, Prompt Engineering, Collaborative Modeling, Explainable AI.

Κεφάλαιο 1 – Εισαγωγή

Η ανάγκη για αυτόματη παραγωγή UML διαγραμμάτων αυξάνεται συνεχώς, καθώς οι ομάδες ανάπτυξης λογισμικού αναζητούν τρόπους να βελτιώσουν την τεκμηρίωση, την κατανόηση και τη συνεργασία κατά την ανάλυση και τον σχεδιασμό συστημάτων. Τα UML διαγράμματα αποτελούν βασικό εργαλείο σε αυτή τη διαδικασία, όμως η δημιουργία τους είναι συχνά μια χρονοβόρα εργασία που απαιτεί και κατάλληλη εμπειρία.

Με την έλευση των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models – LLMs), όπως το GPT, το DeepSeek και το Gemini, παρατηρείται μια ραγδαία πρόοδος στην ικανότητα των συστημάτων Τεχνητής Νοημοσύνης να κατανοούν και να επεξεργάζονται φυσική γλώσσα με τρόπο που προσεγγίζει την ανθρώπινη σκέψη. Τα LLMs είναι μεγάλης κλίμακας νευρωνικά δίκτυα, εκπαιδευμένα σε τεράστιους όγκους δεδομένων κειμένου, τα οποία διαθέτουν τη δυνατότητα παραγωγής, κατανόησης και μετατροπής κειμένου με εντυπωσιακή ευελιξία. Στο πλαίσιο της σχεδίασης λογισμικού, τα μοντέλα αυτά μπορούν να αξιοποιηθούν για την αυτόματη μετάφραση φυσικών περιγραφών σε δομημένα τεχνικά μοντέλα, όπως UML διαγράμματα, γεφυρώνοντας το χάσμα μεταξύ περιγραφικής και τυπικής αναπαράστασης. Η ικανότητά τους να αντλούν συμφραζόμενα, να αναγνωρίζουν οντότητες και σχέσεις και να ακολουθούν οδηγίες σε σύνθετα σενάρια, καθιστά τα LLMs ιδιαίτερα κατάλληλα για εφαρμογές μοντελοποίησης που βασίζονται σε φυσική γλώσσα.

Η παρούσα διπλωματική εργασία συνδυάζει προηγμένες τεχνολογίες Τεχνητής Νοημοσύνης (LLMs) με τεχνικές μοντελοποίησης, προσφέροντας μια πλήρως λειτουργική εφαρμογή ιστού (web application) που μεταφράζει περιγραφές σε φυσική γλώσσα σε UML διαγράμματα. Η εφαρμογή υποστηρίζει την παραγωγή διαγραμμάτων κλάσεων (class), περιπτώσεων χρήσης (use case), ακολουθίας (sequence) και δραστηριοτήτων (activity), με δυνατότητες επιλογής μοντέλου LLM προς χρήση, προσαρμογής μεγέθους διαγράμματος και εξαγωγής διαγράμματος σε PNG, SVG ή ASCII μορφή.

Το βασικό της πλεονέκτημα είναι η ευχρηστία: επιτρέπει σε χρήστες χωρίς εξειδίκευση να δημιουργούν σύνθετα μοντέλα σχεδίασης, μέσω μιας διαδραστικής, επεκτάσιμης και φιλικής προς τον χρήστη γραφικής διεπαφής χρήστη. Επιπλέον, ενσωματώνει δυνατότητες διαχείρισης χρηστών, υποστήριξης πολλαπλών μοντέλων LLM και εξαγωγής διαγραμμάτων σε διάφορες μορφές. Τέλος, επιτρέπει τόσο την παροχή ανατροφοδότησης προς το επιλεγμένο LLM για την βελτίωση των παραγόμενων διαγραμμάτων όσο και την χειρωνακτική τροποποίηση των διαγραμμάτων μέσω της εν κινήσει (on-the-fly) αλλαγής του PlantUML μοντέλου τους. Συνεπώς, η υλοποιημένη εφαρμογή στο πλαίσιο της τρέχουσας διπλωματικής εργασίας είναι τεχνικά ολοκληρωμένη και πρακτικά αξιοποιήσιμη, καλύπτοντας διάφορα σενάρια αλληλεπίδρασης με τον χρήστη και με τα υποστηριζόμενα μοντέλα LLMs.

Το υπόλοιπο μέρος της τρέχουσας αναφοράς οργανώνεται ως εξής:

- Στο **Κεφάλαιο 2** περιγράφεται το θεωρητικό υπόβαθρο σχετικά με τα UML διαγράμματα και τα LLMs.
- Στο **Κεφάλαιο 3** γίνεται ανασκόπηση σχετικών εργασιών.
- Στο **Κεφάλαιο 4** αναλύεται η ανάπτυξη του συστήματος.
- Στο **Κεφάλαιο 5** παρουσιάζεται η εγκατάσταση και χρήση της εφαρμογής.

- Στο **Κεφάλαιο 6** περιλαμβάνεται η αξιολόγηση της εφαρμογής.
- Τέλος, στο **Κεφάλαιο 7** παρουσιάζονται βασικά συμπεράσματα καθώς και προτάσεις για μελλοντική εργασία.

Κεφάλαιο 2 – Υπόβαθρο

2.1 Μοντελοποίηση Συστήματος και UML

Η μοντελοποίηση συστημάτων αποτελεί θεμέλιο λίθο στη διαδικασία ανάπτυξης λογισμικού, καθώς επιτρέπει τη δομημένη κατανόηση, τεκμηρίωση και επικοινωνία των απαιτήσεων, της λειτουργικότητας και της αρχιτεκτονικής ενός συστήματος. Μέσα από διαγράμματα και αφαιρετικές αναπαραστάσεις, ενισχύεται η συνεργασία μεταξύ αναλυτών, αρχιτεκτόνων λογισμικού, προγραμματιστών και τελικών χρηστών. **Η μοντελοποίηση εφαρμόζεται κυρίως κατά τις φάσεις της ανάλυσης απαιτήσεων και της σχεδίασης, όπου απαιτείται σαφής αποτύπωση της λογικής και της ροής του συστήματος.**

Το Unified Modeling Language (UML), το οποίο καθιερώθηκε από τον οργανισμό Object Management Group (OMG), αποτελεί το κυρίαρχο πρότυπο για την οπτική αναπαράσταση συστημάτων λογισμικού. Στηρίζεται στην αντικειμενοστραφή προσέγγιση ανάπτυξης συστημάτων και περιλαμβάνει ένα ευρύ σύνολο διαγραμμάτων, τα οποία χωρίζονται σε δύο βασικές κατηγορίες:

- **Διαγράμματα δομής, όπως:**
  - **Class Diagrams** (Διαγράμματα Κλάσεων): Απεικονίζουν τις κλάσεις του συστήματος, τις ιδιότητές τους και τις σχέσεις μεταξύ τους.
  - **Component Diagrams** (Διαγράμματα Συστατικών): Αναδεικνύουν τα κύρια υποσυστήματα ή συστατικά ενός συστήματος και τις μεταξύ τους συνδέσεις μέσω σχετικών προσφερόμενων και απαιτούμενων διεπαφών.
- **Διαγράμματα συμπεριφοράς, όπως:**
  - **Use Case Diagrams** (Διαγράμματα Περιπτώσεων Χρήσης): Περιγράφουν τις αλληλεπιδράσεις χρηστών με το σύστημα στο πλαίσιο μιας ή περισσοτέρων περιπτώσεων χρήσης. Οι αλληλεπιδράσεις καταγράφονται σε ένα αρκετά υψηλό επίπεδο όπου μία (συνήθως) ή περισσοτέρες αλληλεπιδράσεις αντιστοιχούν σε μια περίπτωση χρήσης.
  - **Sequence Diagrams** (Διαγράμματα Ακολουθίας): Καταγράφουν τη ροή μηνυμάτων μεταξύ αντικειμένων/συστατικών/υπο-συστημάτων ενός συστήματος λογισμικού με χρονική σειρά στο πλαίσιο περαίωσης μιας περίπτωσης χρήσης. Συνήθως, περιλαμβάνουν και κάποιον χρήστη, ο οποίος εκκινεί τη ροή με ένα πρώτο μήνυμα. Μπορεί να εμπεριέχουν και ειδικούς τελεστές ροής (flow operators) για την κάλυψη πολύπλοκων ροών μηνυμάτων.
  - **Activity Diagrams** (Διαγράμματα Δραστηριοτήτων): Χρησιμοποιούνται για την αποτύπωση ροών εργασιών (workflows) ή επιχειρησιακών διαδικασιών (business processes), καταγράφοντας τη ροή ελέγχου (control flow) (δηλ. εκτέλεσης δραστηριοτήτων) και ροή δεδομένων (data flow) που πραγματοποιούνται στο πλαίσιο αυτών των ροών εργασιών / διαδικασιών.

Παρόμοια με τα διαγράμματα ακολουθίας, μπορούν να καλύπτουν τον τρόπο περαίωσης μιας περίπτωσης χρήσης.

Στην πράξη, η επιλογή του κατάλληλου διαγράμματος εξαρτάται από τη φάση ανάπτυξης (π.χ. ανάλυση απαιτήσεων, σχεδίαση) και τον στόχο του κάθε μοντέλου. Για παράδειγμα, κατά την ανάλυση απαιτήσεων χρησιμοποιούνται συχνά διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams) για να αποδοθούν οι βασικές αλληλεπιδράσεις μεταξύ χρηστών και συστήματος, ενώ στη φάση της σχεδίασης προτιμώνται διαγράμματα ακολουθίας (Sequence Diagrams) ή διαγράμματα δραστηριοτήτων (Activity Diagrams) για την αποτύπωση της δυναμικής συμπεριφοράς και της ροής των ενεργειών του συστήματος.

Η παραγωγή διαγραμμάτων είναι αρκετά σημαντική στο πλαίσιο διαδικασιών ανάπτυξης που οδηγούνται από πλάνο (plan-based development processes) καθώς και σε προσεγγίσεις ανάπτυξης που οδηγούνται από μοντέλα (model-driven development approaches). Ειδικά, στην περίπτωση των τελευταίων προσεγγίσεων, είναι δυνατή η αυτόματη παραγωγή του κώδικα της εφαρμογής από τα παραγόμενα διαγράμματα/μοντέλα του συστήματος.

Υπάρχουν πολλά εργαλεία που υποστηρίζουν τη δημιουργία UML διαγραμμάτων, με χαρακτηριστικά που ποικίλουν ανάλογα με τις ανάγκες και την εξειδίκευση του χρήστη. Οι βασικές λειτουργίες που υποστηρίζονται από αυτά (όπως δημιουργία/επεξεργασία διαγραμμάτων, εξαγωγή σε μορφές εικόνας, συνεργασία σε πραγματικό χρόνο, αποθήκευση στο υπολογιστικό νέφος (cloud)) καθορίζουν σε μεγάλο βαθμό την καταλληλότητα κάθε εργαλείου ως προς τις απαιτήσεις και προτιμήσεις του χρήστη.

Ο παρακάτω πίνακας συγκρίνει ορισμένα από τα πιο διαδεδομένα εργαλεία με βάση κριτήρια που σχετίζονται άμεσα με την παρούσα εργασία: τα είδη UML διαγραμμάτων που υποστηρίζονται, τον τύπο του προσφερόμενου προγράμματος επεξεργασίας διαγραμμάτων (editor) (γραφικός ή βασισμένος σε κείμενο), τις υποστηριζόμενες λειτουργίες, την πολιτική αδειοδότησης, τη δυνατότητα αποθήκευσης (διαγραμμάτων) στο νέφος και τις ενσωματώσεις (integrations) με άλλα εργαλεία.

Πίνακας 1. Σύγκριση διαδεδομένων εργαλείων μοντελοποίησης UML διαγραμμάτων

Εργαλείο	Είδη UML διαγραμμάτων	Τύπος editor	Λειτουργίες	Άδεια	Υποστήριξη cloud	Ενσωματώσεις
Lucidchart	Use Case, Class, Sequence, Activity κ.ά.	Γραφικός	Δημιουργία, επεξεργασία, εξαγωγή διαγραμμάτων, συνεργατική μοντελοποίηση	Εμπορικό (freemium)	✅	✅ (Jira, GitHub, Slack)
Draw.io	Πολλά είδη UML	Γραφικός	Βασική σχεδίαση, εξαγωγή	Ανοιχτό / Δωρεάν	✅ (Google Drive)	❌
PlantUML	Όλα τα είδη UML	Κειμενικός	Μοντελοποίηση διαγραμμάτων μέσω μοντέλων/κώδικα (diagrams-as-code), εξαγωγή διαγράμματος σε PNG/SVG/ASCII	Ανοιχτό	❌	✅ (CI/CD, IDE plugins, GitHub/GitLab CI, Markdown, wikis)
StarUML	UML και SysML	Γραφικός	Μοντελοποίηση UML/SysML & ERD, reverse engineering	Εμπορικό	❌	❌

Η παρούσα εφαρμογή αξιοποιεί το PlantUML ως έναν backend μηχανισμό απεικόνισης διαγραμμάτων μέσω των κειμενικών μοντέλων περιγραφής τους, τα οποία μπορούν να τροποποιούνται δυναμικά από τον χρήστη, παρέχοντας ένα ευέλικτο και επεκτάσιμο περιβάλλον για τη μοντελοποίηση λογισμικού. Η επιλογή του PlantUML έναντι άλλων εργαλείων βασίστηκε στη δυνατότητά του για "diagrams as code", στην ευκολία ενσωμάτωσής του σε αγωγούς (pipelines) CI/CD (Continuous Integration / Continuous Delivery) και άλλα εργαλεία ανάπτυξης λογισμικού, καθώς και στη συμβατότητά του με διάφορες μορφές εξαγωγής και αυτοματοποίησης

Αν και η κλασική προσέγγιση DevOps εστιάζει στη συχνή παράδοση κώδικα και στην ελαχιστοποίηση της τεκμηρίωσης, υπάρχουν περιπτώσεις όπου η χρήση UML διαγραμμάτων σε μορφή κώδικα (*diagrams as code*) (όπως υποστηρίζεται από το PlantUML), ενσωματωμένων σε CI/CD αγωγούς, έχει ουσιαστικό νόημα. Τέτοιες περιπτώσεις περιλαμβάνουν την αυτόματη δημιουργία ή ενημέρωση διαγραμμάτων σε συγχρονισμό με αλλαγές στον κώδικα, την τεκμηρίωση ως κώδικας (*documentation-as-code*) μέσω εργαλείων όπως **Markdown και AsciiDoc** (*μορφές ελαφριάς σήμανσης για τεχνική τεκμηρίωση, οι οποίες υποστηρίζονται από πολλά συστήματα wiki, CI pipelines και static site generators*), την οπτικοποίηση διαφορών σε κλαδιά (branches), καθώς και την παραγωγή συμβατής τεκμηρίωσης στο πλαίσιο απαιτήσεων συμμόρφωσης. Σε αυτό το πλαίσιο, η παρούσα εργασία προσφέρει δυνατότητες που ενισχύουν την ενσωμάτωσή της σε DevOps διαδικασίες, ιδιαίτερα σε έργα που υιοθετούν μοντέλα συνεργατικής σχεδίασης και αυτόματης τεκμηρίωσης.

2.2 Τεχνητή Νοημοσύνη και Μεγάλα Γλωσσικά Μοντέλα

Η **Τεχνητή Νοημοσύνη (ΤΝ)** (*Artificial Intelligence – AI*) αποτελεί τον επιστημονικό κλάδο της πληροφορικής που ασχολείται με τη δημιουργία συστημάτων ικανών να εκτελούν εργασίες, οι οποίες απαιτούν ανθρώπινη νοημοσύνη, όπως η λήψη αποφάσεων, η μάθηση από εμπειρία, η κατανόηση φυσικής γλώσσας και η αναγνώριση προτύπων. Τα βασικά είδη της ΤΝ περιλαμβάνουν:

- **Επιβλεπόμενη μάθηση (Supervised Learning):** Το σύστημα μαθαίνει από δεδομένα που περιέχουν και τα επιθυμητά αποτελέσματα (μέσω ετικετών - labels), ώστε να είναι σε θέση να πραγματοποιεί προβλέψεις ή ταξινομήσεις.

- **Μη επιβλεπόμενη μάθηση (Unsupervised Learning):** Το σύστημα προσπαθεί να ανακαλύψει πρότυπα ή ομάδες μέσα από αδόμητα δεδομένα χωρίς προκαθορισμένες εξόδους.

- **Ενισχυτική μάθηση (Reinforcement Learning):** Το σύστημα μαθαίνει μέσω αλληλεπίδρασης με το περιβάλλον του, λαμβάνοντας ενισχύσεις (rewards) ή τιμωρίες για τις ενέργειές του από το περιβάλλον αυτό.

Μια από τις ταχύτερα εξελισσόμενες περιοχές της ΤΝ είναι η Generative AI (Παραγωγική ΤΝ), με βασικό υποκλάδο τα Μοντέλα Μεγάλης Γλώσσας (Large Language Models - LLMs). Τα LLMs, όπως το GPT-4 της OpenAI, το Gemini της Google και το DeepSeek, βασίζονται σε αρχιτεκτονικές μετασχηματιστών (transformers) και εκπαιδεύονται πάνω σε τεράστιες ποσότητες δεδομένων. Οι αρχιτεκτονικές μετασχηματιστών αποτελούν ένα είδος νευρωνικού δικτύου που βασίζεται σε μηχανισμούς προσοχής (attention), επιτρέποντας στο μοντέλο να εστιάζει δυναμικά στα πιο σημαντικά μέρη της εισόδου για κάθε έξοδο, με υψηλή παράλληλη επεξεργασία και αποτελεσματικότητα στην εκμάθηση σχέσεων μεγάλης εμβέλειας μέσα σε ακολουθίες δεδομένων.

Οι δυνατότητες των LLMs επεκτείνονται στη μετάφραση φυσικής γλώσσας σε δομημένο κείμενο, ψευδοκώδικα ή πραγματικό κώδικα, στην αυτόματη παραγωγή τεκμηρίωσης για APIs, καθώς και στην εξαγωγή ενδιάμεσων αναπαραστάσεων – όπως μοντέλα PlantUML – από περιγραφικά σενάρια. Παράλληλα, υποστηρίζουν τον αυτόματο εντοπισμό και τη διόρθωση σφαλμάτων σε κώδικα. Πέραν αυτών, τα LLMs μπορούν να χρησιμοποιηθούν για **κατανόηση διαγραμμάτων (diagram recognition/understanding)** και **μετατροπή από διαγράμματα σε τυπικά μοντέλα (diagram-to-model conversion)**, δηλαδή την εξαγωγή τυπικής φυσικής περιγραφής ή ψευδοκώδικα (δηλ. του σχετικού μοντέλου) από υπάρχοντα UML διαγράμματα (που βρίσκονται πχ. σε μορφή εικόνας). Επίσης, μπορούν να συνεισφέρουν στον **εντοπισμό λογικών ή συντακτικών σφαλμάτων σε διαγράμματα UML**, καθώς και στην **αυτόματη πρόταση διορθώσεων** στα διαγράμματα αυτά. Τέτοιες λειτουργίες επεκτείνουν τις παραδοσιακές δυνατότητες εργαλείων μοντελοποίησης και υποδεικνύουν ότι τα LLMs μπορούν να λειτουργήσουν ως **έξυπνοι βοηθοί μηχανικού λογισμικού (software engineer assistants)**.

Η παρούσα εφαρμογή αξιοποιεί την ισχύ των LLMs ώστε να γεφυρώσει το χάσμα μεταξύ ανθρώπινης περιγραφής και τεχνικής απεικόνισης, προσφέροντας ένα εργαλείο που ενσωματώνει την ευκολία της φυσικής γλώσσας με τη δύναμη της αυτοματοποιημένης μοντελοποίησης. Ειδικότερα, υποστηρίζεται η παραγωγή ενδιάμεσων αναπαραστάσεων σε γλώσσα PlantUML από περιγραφικά σενάρια σε φυσική γλώσσα, οι οποίες στη συνέχεια μετατρέπονται αυτόματα σε UML διαγράμματα (κλάσεων, ακολουθίας, περιπτώσεων χρήσης και δραστηριοτήτων).

Κεφάλαιο 3 – Σχετικές Εργασίες

3.1 Model Generation with LLMs: From Requirements to UML Sequence Diagrams

Alessio Ferrari, Sallam Abualhaija, Chetan Arora (2024)

Η εργασία αυτή μελετά την ικανότητα του **ChatGPT** να δημιουργεί **UML διαγράμματα ακολουθίας (sequence diagrams)** με βάση περιγραφές απαιτήσεων σε φυσική γλώσσα. Οι συγγραφείς δημιούργησαν ένα σύνολο δεδομένων με 28 έγγραφα απαιτήσεων και ζήτησαν από το LLM (**Chat GPT-4)** μοντέλο να παράγει τα αντίστοιχα διαγράμματα ακολουθίας. Ακολούθησε αξιολόγηση από ειδικούς για την καταγραφή της ορθότητας, πληρότητας και ευχρηστίας των παραγόμενων διαγραμμάτων. Η μελέτη επιβεβαίωσε πως τα LLMs έχουν τη δυνατότητα να παράγουν τυπικά σωστά διαγράμματα, ικανά να απεικονίζουν βασικές αλληλεπιδράσεις.

Ωστόσο, παρατηρήθηκαν αδυναμίες κυρίως στην πληρότητα της πληροφορίας, με το μοντέλο να χάνει κρίσιμες λεπτομέρειες από τις απαιτήσεις. Σύμφωνα με τους συγγραφείς, αυτό οφείλεται στο γεγονός ότι τα LLMs συχνά **εστιάζουν στις πιο εμφανείς και γενικές πληροφορίες**, παραβλέποντας επιμέρους λεπτομέρειες όταν αυτές δεν εμφανίζονται με σαφή και ρητό τρόπο στο αρχικό κείμενο. Επιπλέον, η ορθότητα εξαρτάται σημαντικά από την σαφήνεια και ποιότητα της εισόδου, ενώ η **ευχρηστία** των παραγόμενων διαγραμμάτων αξιολογήθηκε ως καλή, ιδίως όταν πρόκειται για απλά σενάρια και σαφώς δομημένες απαιτήσεις.

Στα θετικά, η μελέτη προσφέρει μεθοδική προσέγγιση αξιολόγησης και δείχνει ότι ακόμα και χωρίς εκπαίδευση σε UML, τα LLMs παρουσιάζουν αξιοσημείωτη ικανότητα σύνθεσης. Στα αρνητικά, η ανάγκη για ανθρώπινη επιμέλεια και η μη επαναληψιμότητα σε πιο σύνθετα σενάρια (δηλ. η παραγωγή κάθε φορά διαφορετικών διαγραμμάτων από τις ίδιες περιγραφές απαιτήσεων) είναι ζητήματα που αναδεικνύονται.

3.1.2 From Image to UML: Image-Based UML Diagram Generation Using LLMs

Aaron Conrardy, Jordi Cabot (2024)

Η εργασία αυτή εξετάζει μια λιγότερο συμβατική προσέγγιση: την αναγνώριση UML διαγραμμάτων από εικόνες και την ανακατασκευή τους σε τυπική μορφή (π.χ. PlantUML) χρησιμοποιώντας LLMs. Οι ερευνητές αξιοποίησαν **τεχνολογία OCR** για την εξαγωγή του κειμενικού περιεχομένου και των βασικών συμβολισμών από τις εικόνες των διαγραμμάτων (όπως ονόματα κλάσεων, συσχετίσεις, ετικέτες, είδη γραμμών κ.λπ.). Στη συνέχεια, το εξαγόμενο αυτό περιεχόμενο οργανώνεται σε δομημένη μορφή και παρέχεται ως είσοδος σε ένα LLM (όπως το GPT-4), το οποίο αναλαμβάνει να **ερμηνεύσει και να μετασχηματίσει** τις πληροφορίες σε περιγραφή PlantUML. Ο στόχος ήταν να κατασκευαστεί αυτόματα ένα συντακτικά σωστό μοντέλο UML, το οποίο μπορεί να επεξεργαστεί περαιτέρω, να αναλυθεί ή να μετατραπεί σε εναλλακτικές μορφές

Η προσέγγιση παρουσιάζει μεγάλο ενδιαφέρον, καθώς συνδυάζει πολυτροπικά δεδομένα (εικόνα & κείμενο), ωστόσο απαιτεί σημαντική ανθρώπινη επίβλεψη για τη διόρθωση ανακριβειών (όπως λανθασμένη αναγνώριση ονομάτων κλάσεων, παράλειψη βελών συσχέτισης ή ανασφαλή εντοπισμό πληθυκοτήτων (cardinalities)). Αν και η ιδέα είναι πρωτότυπη, το σύστημα δυσκολεύεται με πολύπλοκες διατάξεις ή χειρόγραφες UML αναπαραστάσεις. Πλεονεκτήματα της εργασίας είναι η καινοτομία και η χρησιμότητα σε αναστροφή παλαιών σχεδίων UML, ενώ τα μειονεκτήματα περιλαμβάνουν τη χαμηλή ακρίβεια και τη χαμηλή γενίκευση σε διαφορετικούς τύπους εικόνων.

3.1.3 UML Code Generation from Diagram Images Using Multimodal LLMs

Averi Bates et al. (2024)

Αυτή η εργασία προτείνει μια προσέγγιση παραγωγής UML κώδικα από εικόνες διαγραμμάτων, χρησιμοποιώντας **πολυτροπικά LLMs** που έχουν τη δυνατότητα να επεξεργάζονται ταυτόχρονα εικόνα και κείμενο. Οι συγγραφείς πραγματοποίησαν **fine-tuning** σε ένα υπάρχον πολυτροπικό LLM (το GPT-4V), εκπαιδεύοντάς το περαιτέρω σε ένα εξειδικευμένο σύνολο δεδομένων που περιλάμβανε εικόνες UML διαγραμμάτων ακολουθίας και τις αντίστοιχες αναπαραστάσεις τους σε κώδικα PlantUML. Το **fine-tuning** είναι η διαδικασία όπου ένα ήδη εκπαιδευμένο μοντέλο υποβάλλεται σε επιπλέον εκπαίδευση σε ειδικά δεδομένα για να βελτιώσει την απόδοσή του σε μια συγκεκριμένη εφαρμογή ή τύπο περιεχομένου.

Η μεθοδολογία αξιολογήθηκε ως προς την ακρίβεια και τη δυνατότητα γενίκευσης των παραγόμενων αποτελεσμάτων σε **νέα, άγνωστα είδη διαγραμμάτων**. Εκτός από διαγράμματα ακολουθίας, εξετάστηκαν επίσης **διαγράμματα κλάσεων** και **διαγράμματα δραστηριοτήτων**. Η αξιολόγηση πραγματοποιήθηκε **με μετρικές ακρίβειας και συντακτικής ορθότητας**, χωρίς όμως την εμπλοκή εξωτερικών ειδικών αξιολογητών, όπως συνέβη στην εργασία των Ferrari et al. (2024). Αντίθετα, η μέτρηση της ποιότητας έγινε μέσω αυτόματης σύγκρισης εξόδων με προκαθορισμένες αναπαραστάσεις-στόχους (ground truth).

Το βασικό πλεονέκτημα της προσέγγισης είναι η δυνατότητα αξιοποίησης υλικού που δεν έχει αρχικά κατασκευαστεί με UML εργαλεία, δηλαδή **εικόνων διαγραμμάτων που προέρχονται από εξωτερικές ή παλαιότερες πηγές**. Το βασικό μειονέκτημα εντοπίζεται στην **ευαισθησία της διαδικασίας σε σφάλματα σύνταξης κατά τη μετατροπή του διαγράμματος σε PlantUML**, καθώς μικρές οπτικές αποκλίσεις ή ασάφειες στην εικόνα μπορεί να οδηγήσουν σε λανθασμένο κώδικα (π.χ. ελλείψεις κλεισίματος blocks, εσφαλμένοι τύποι συσχετίσεων ή ασύμβατα σύμβολα). Οι συγγραφείς αποδίδουν αυτά τα λάθη στον **συνδυαστικό θόρυβο της OCR εξαγωγής και της γλωσσικής ερμηνείας από το LLM**, ειδικά όταν δεν υπάρχει τυποποιημένη μορφοποίηση στην είσοδο.

Η προσέγγιση χαρακτηρίζεται ως υποσχόμενη κυρίως για την **αναστροφή παλαιών UML διαγραμμάτων**, όπου δεν είναι διαθέσιμο το αρχικό αρχείο μοντελοποίησης αλλά μόνο η τελική απεικόνιση (σε μορφή εικόνας). Αντί της ασαφούς αναφοράς σε "μετατροπή αρχείων", ο στόχος της μεθόδου είναι η **μετατροπή εικόνων UML διαγραμμάτων σε επεξεργάσιμες αναπαραστάσεις** τύπου PlantUML, διευκολύνοντας την τεκμηρίωση, την ανάλυση ή τη μετεξέλιξή τους σε νέα συστήματα.

3.2 Σύγκριση με την παρούσα εργασία

Η παρούσα διπλωματική εργασία διαφοροποιείται σημαντικά από τις παραπάνω ερευνητικές προσεγγίσεις, τόσο ως προς την **εύρος υποστήριξης διαγραμμάτων** όσο και ως προς τη **προσφερόμενη λειτουργικότητα**, όπως φαίνεται και από τον Πίνακα 2

Σε αντίθεση με τις περισσότερες ερευνητικές εργασίες που επικεντρώνονται σε ένα μόνο είδος διαγράμματος (κυρίως ακολουθίας ή κλάσεων), η προτεινόμενη εφαρμογή υποστηρίζει **τέσσερις** διαφορετικούς τύπους UML διαγραμμάτων: Κλάσεων, Ακολουθίας, Περιπτώσεων Χρήσης και Δραστηριοτήτων. Επιπλέον, ενώ αρκετές εργασίες βασίζονται σε **στατικά prompts** (προτροπές), δηλαδή απλές, μονοστροφικές ερωτήσεις προς το LLM χωρίς δυνατότητα ανατροφοδότησης, η προτεινόμενη εφαρμογή ενσωματώνει **διαδραστική ανατροφοδότηση** (interactive feedback). Παράλληλα, δίνεται η δυνατότητα στον χρήστη να **τροποποιήσει το παραγόμενο PlantUML μοντέλο**, επηρεάζοντας δυναμικά το τελικό διάγραμμα. Με αυτόν τον τρόπο, η αλληλεπίδραση δεν είναι παθητική ή εφάπαξ, αλλά εξελικτική και προσαρμοστική, γεγονός που διαφοροποιεί ουσιαστικά την προτεινόμενη εφαρμογή από τις υφιστάμενες ερευνητικές προσεγγίσεις.

Η δυνατότητα επιλογής μεγέθους διαγράμματος και μορφοποίησης εξαγωγής (PNG, SVG, ASCII) ενισχύει τη χρηστικότητα της εφαρμογής, καθώς επιτρέπει την προσαρμογή της εξόδου στον εκάστοτε σκοπό: παρουσίαση, ενσωμάτωση σε τεκμηρίωση, χρήση σε περιβάλλοντα χωρίς υποστήριξη εικόνας κ.ά. Με αυτόν τον τρόπο, η εφαρμογή προσφέρει μεγαλύτερη λειτουργική κάλυψη από τα υπάρχοντα

ερευνητικά πρωτότυπα και μπορεί να χρησιμοποιηθεί άμεσα σε πραγματικά έργα λογισμικού.

Πίνακας 2. Σύγκριση με Σχετικές Εργασίες

Εργασία	Τύποι Διαγραμμάτων	LLM(s)	Διαδραστικότητα	Τροποποίηση Μοντέλου	Παρατηρήσεις
Ferrari et al. (2024)	Sequence	ChatGPT	✗	✗	Καλή τυπική δομή, χαμηλή πληρότητα
Conrardy & Cabot (2024)	Διάφορα (εικόνα → diagram code)	GPT-4	✗	✗	Καινοτόμος προσέγγιση βασιζόμενη σε εικόνες
Bates et al. (2024)	Sequence (εικόνα → diagram code)		✗	✗	Πολυτροπική χρήση LLMs
Παρούσα εργασία	**Class, Sequence, Use Case, Activity**	**GPT 4, Gemini1.5 Flash, DeepSeek Coder v2**	✓	✓	Πλήρες σύστημα με γραφική διεπαφή χρήστη, διαχείριση χρηστών, δυνατότητες εξαγωγής σε διαφορετικές μορφοποιήσεις εικόνων

Κεφάλαιο 4 – Ανάπτυξη Συστήματος

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε ακολουθώντας μια μεθοδολογική προσέγγιση με έμφαση στη λειτουργικότητα, την επεκτασιμότητα και την εμπειρία χρήστη. Ο στόχος ήταν η δημιουργία μιας **εφαρμογής ιστού** (web application), η οποία επιτρέπει τη **δημιουργία UML διαγραμμάτων με τη βοήθεια Μεγάλων**

Γλωσσικών Μοντέλων (LLMs), προσφέροντας στον χρήστη ένα **απλό και ευέλικτο περιβάλλον** αλληλεπίδρασης.

Για την υλοποίηση χρησιμοποιήθηκε το **πλαίσιο Django** για το backend, εξασφαλίζοντας δομή, ασφάλεια και επεκτασιμότητα, ενώ το frontend βασίστηκε σε **HTML/CSS και JavaScript**, με ενσωμάτωση δυναμικών στοιχείων και αποκριτικού (responsive) σχεδιασμού. Η εφαρμογή υποστηρίζει **εγγραφή, είσοδο και διαχείριση χρηστών**, επικοινωνεί με **διάφορα LLMs μέσω API** και επιτρέπει την **παραγωγή, επεξεργασία και εξαγωγή UML διαγραμμάτων** μέσω του εργαλείου PlantUML.

Η διαδικασία ανάπτυξης οργανώθηκε σε βασικά στάδια, τα οποία υλοποιήθηκαν σειριακά με βάση το μοντέλο του καταρράκτη (waterfall model). Καθένα από αυτά τα στάδια περιγράφεται αναλυτικά στις επόμενες ενότητες μαζί με τα κύρια αποτελέσματα που παρήγαγε:

- **Ανάλυση Απαιτήσεων**: Ορισμός βασικών και έξτρα λειτουργιών του συστήματος καθώς και των μη λειτουργικών του απαιτήσεων.
- **Σχεδίαση**: Δημιουργία UML διαγραμμάτων που αποτυπώνουν τη δομή και τη ροή λειτουργίας (δηλ. την συμπεριφορά) του συστήματος.
- **Υλοποίηση**: Ανάλυση των τεχνολογιών, της αρχιτεκτονικής και της δομής του κώδικα.

4.1 Ανάλυση Απαιτήσεων

Η ανάλυση απαιτήσεων αποτελεί ένα από τα σημαντικότερα στάδια κατά την ανάπτυξη συστημάτων λογισμικού, καθώς προσδιορίζει με ακρίβεια τις δυνατότητες που οφείλει να υποστηρίζει το τελικό σύστημα. Στο πλαίσιο της παρούσας εργασίας, οι απαιτήσεις διακρίνονται σε **λειτουργικές** και **μη λειτουργικές**, οι οποίες με τη σειρά τους κατηγοριοποιούνται σε **υποχρεωτικές** και **προαιρετικές**. Ο χαρακτηρισμός μιας απαίτησης ως υποχρεωτικής δηλώνει ότι είναι απαραίτητη για τη βασική λειτουργικότητα του συστήματος, ενώ οι προαιρετικές απαιτήσεις επεκτείνουν ή ενισχύουν τη λειτουργικότητα χωρίς να είναι απαραίτητες για την ορθή λειτουργία της εφαρμογής - συνεπώς, δεν είναι πάντοτε υποχρεωτική η υλοποίησή τους αλλά ενδέχεται να παρέχουν προστιθέμενη αξία σε μια εφαρμογή. Η διαφοροποίηση αυτή αποτυπώνεται τόσο δομικά όσο και με κατάλληλη χρήση γλώσσας κατά την παρουσίαση των επιμέρους απαιτήσεων.

4.1.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις της εφαρμογής οργανώνονται σε τρεις βασικές κατηγορίες: **(α) Παροχή εισόδου χρήστη**, **(β) Παραγωγή διαγράμματος**, και **(γ) Εμφάνιση και εξαγωγή του διαγράμματος**. Κάθε απαίτηση φέρει σύντομο τίτλο και προσδιορίζεται ως **υποχρεωτική** ή **προαιρετική** ενώ ακολουθεί μια σύντομη περιγραφική της ανάλυση.

(α) Παροχή εισόδου χρήστη

- **Εισαγωγή περιγραφής προβλήματος *(Υποχρεωτική)***: Το σύστημα θα πρέπει να επιτρέπει στον χρήστη να παρέχει μια περιγραφή του προβλήματος ή του συστήματος (λογισμικού προς μοντελοποίηση) σε φυσική γλώσσα, μέσω ειδικής φόρμας.

- **Επιλογή τύπου διαγράμματος (Υποχρεωτική)**: Ο χρήστης πρέπει να μπορεί να επιλέγει τον τύπο UML διαγράμματος προς παραγωγή (Class, Use Case, Sequence, Activity). Επιτρέπεται μόνο ένας τύπος ανά υποβολή. *Σημείωση:* Οι εγγεγραμμένοι χρήστες έχουν πρόσβαση σε όλους τους τύπους διαγραμμάτων, ενώ οι επισκέπτες μπορούν να επιλέξουν μόνο Use Case ή Activity διαγράμματα για δοκιμή της λειτουργικότητας (της εφαρμογής).

- **Επιλογή LLM *(Υποχρεωτική)***: Ο χρήστης επιλέγει το επιθυμητό LLM (π.χ. GPT, Gemini, DeepSeek). Το σύστημα θα πρέπει να ζητά και να αποθηκεύει **API access token** για το αντίστοιχο/επιλεγμένο LLM με ασφαλή τρόπο, κατά την πρώτη επιλογή ή χρήση του κάθε μοντέλου από τον χρήστη. Προφανώς, η δυνατότητα αποθήκευσης αφορά κυρίως τους εγγεγραμμένους χρήστες του συστήματος.

- **Παροχή παραμέτρων διαμόρφωσης *(Προαιρετική)***: Ο χρήστης μπορεί να ορίσει επιπλέον παραμέτρους, όπως το μέγεθος του διαγράμματος (π.χ. “Μικρό” ή “Μεγάλο”), ώστε να επηρεάσει το περιεχόμενο του prompt που θα σταλεί στο LLM και άρα και το παραγόμενο διάγραμμα.

(β) Παραγωγή διαγράμματος

- **Διαμόρφωση prompt *(Υποχρεωτική)***: Το σύστημα θα πρέπει να διαμορφώνει κατάλληλα το prompt χρησιμοποιώντας κατάλληλα prompt patterns (μοτίβα προτροπής), συνδυάζοντας την περιγραφή του χρήστη και τις παραμέτρους διαμόρφωσης, ώστε να παράγει συμβατή είσοδο για το LLM.

- **Αποστολή prompt σε LLM και λήψη αποτελέσματος *(Υποχρεωτική)***: Το διαμορφωμένο prompt θα πρέπει να αποστέλλεται στο επιλεγμένο LLM μέσω κατάλληλου API και να αναμένεται έξοδος σε μορφή PlantUML.

- **Έλεγχος συντακτικής εγκυρότητας (Υποχρεωτική):** Το σύστημα θα πρέπει να ελέγχει αν η έξοδος του LLM είναι συντακτικά έγκυρη σε γλώσσα PlantUML. Σε περίπτωση σφάλματος, το σύστημα θα πρέπει να επιχειρεί αυτόματα έως και δύο (2) επιπλέον προσπάθειες παραγωγής έγκυρου διαγράμματος πριν ενημερώσει τον χρήστη για το σφάλμα. Εφόσον αποτύχουν και οι τρεις (3) προσπάθειες, το σύστημα δεν αποστέλλει την έξοδο στον renderer αλλά εμφανίζει σχετικό μήνυμα λάθους στον χρήστη. Το μήνυμα σφάλματος προέρχεται απευθείας από την PlantUML και συχνά παρέχει ένδειξη για τη θέση ή το είδος του προβλήματος (σύνταξης), δίνοντας στον χρήστη την ευκαιρία να κατανοήσει τι μπορεί να πήγε λάθος. Ο χρήστης μπορεί στη συνέχεια να επιλέξει επαναπροσπάθεια, τροποποίηση της αρχικής περιγραφής (prompt), αλλαγή παραμέτρων ή εναλλαγή του μοντέλου LLM, αξιοποιώντας τη σχετική ανατροφοδότηση.

(γ) Εμφάνιση και εξαγωγή διαγράμματος

- **Οπτική αναπαράσταση / εμφάνιση διαγράμματος** *(Υποχρεωτική)*: Αν το αποτέλεσμα είναι έγκυρο, το σύστημα θα πρέπει να αποδώσει το αντίστοιχο διάγραμμα γραφικά.

- **Εξαγωγή διαγράμματος** *(Υποχρεωτική)*: Το σύστημα θα παρέχει τη δυνατότητα εξαγωγής του διαγράμματος σε μία από τις ακόλουθες μορφές: PNG, SVG ή ASCII. Ο χρήστης καθώς και ο επισκέπτης θα επιλέγει μία μόνο μορφή ανά λήψη και αυτή θα αποθηκεύεται στον φάκελο με τις λήψεις στο τοπικό του σύστημα.

4.1.2 Επιπρόσθετες Λειτουργίες / Επεκτάσεις

Οι παρακάτω απαιτήσεις δεν είναι απαραίτητες για την ελάχιστη λειτουργικότητα του συστήματος, αλλά ενισχύουν σημαντικά τη χρηστικότητα, την επεκτασιμότητα και την εμπειρία χρήστη. Ομαδοποιούνται σε δύο βασικές κατηγορίες: **Διαχείριση Χρηστών** και **Εκλέπτυνση Διαγραμμάτων**.

Διαχείριση Χρηστών

Εγγραφή χρήστη *(Προαιρετική)*
Το σύστημα θα πρέπει να επιτρέπει τη δημιουργία νέου λογαριασμού χρήστη μέσω φόρμας εγγραφής, με πεδία όπως όνομα χρήστη, email και κωδικό πρόσβασης.

Σύνδεση / Αποσύνδεση *(Προαιρετική)*
Το σύστημα θα πρέπει να επιτρέπει σε ήδη εγγεγραμμένους χρήστες να συνδεθούν με τα στοιχεία τους καθώς και να αποσυνδεθούν με ασφάλεια.

Επεξεργασία προφίλ *(Προαιρετική)*
Οι χρήστες θα πρέπει να μπορούν να ενημερώνουν τα προσωπικά τους στοιχεία καθώς και τον κωδικό πρόσβασής τους.

Ανάκτηση κωδικού πρόσβασης (Προαιρετική)
Σε περίπτωση απώλειας κωδικού, το σύστημα θα πρέπει να υποστηρίζει ανάκτηση μέσω απάντησης σε προκαθορισμένη ερώτηση ασφαλείας. Ο χρήστης εισάγει τη σωστή απάντηση, και εφόσον αυτή επαληθευτεί, το σύστημα εμφανίζει φόρμα αλλαγής κωδικού, όπου ο χρήστης καλείται να εισαγάγει τον νέο κωδικό πρόσβασης εις διπλούν για επιβεβαίωση. Ο νέος κωδικός ελέγχεται ως προς την εγκυρότητα και την ισχύ του (π.χ. μήκος, σύνθεση). Εφόσον ο έλεγχος περάσει επιτυχώς, ο νέος κωδικός κατακερματίζεται (hashed) και αποθηκεύεται με ασφάλεια στην υποκείμενη βάση δεδομένων, αντικαθιστώντας τον παλαιό.

Αίτηση διαγραφής λογαριασμού *(Προαιρετική)*
Ο χρήστης μπορεί να υποβάλει αίτημα για οριστική διαγραφή του λογαριασμού του. Η αίτηση μπορεί να εγκριθεί από διαχειριστή ή να εκτελεστεί αυτόματα μετά από 3 ημέρες.

Εκλέπτυνση Διαγραμμάτων

Επεξεργασία μοντέλου διαγράμματος (PlantUML) ***(Προαιρετική)*** Το σύστημα θα πρέπει να παρέχει ενσωματωμένο editor, μέσω του οποίου ο χρήστης μπορεί να επεξεργάζεται απευθείας το (κειμενικό) μοντέλο PlantUML που παρήχθη από το LLM, ώστε να διορθώσει ή να επεκτείνει το διάγραμμα.

4.1.3 Μη Λειτουργικές Απαιτήσεις

Η εφαρμογή οφείλει να πληροί επίσης τις ακόλουθες προδιαγραφές ποιότητας και ασφάλειας:

Φιλική και αποκριτική (responsive) διεπαφή χρήστη, συμβατή με πολλαπλές συσκευές (PC, mobile, tablet).
Διαχωρισμένη αρχιτεκτονική Frontend - Backend, με παροχή RESTful API (μέσω Swagger / OpenAPI) από το backend προς εκμετάλλευση από το frontend. Το RESTful API θα πρέπει να είναι τουλάχιστον επιπέδου 2 με βάση το μοντέλο ωριμότητας του Richardson.
Ασφάλεια: Υποστήριξη αυθεντικοποίησης μέσω tokens, χρήση μοντέλου RBAC για τον έλεγχο πρόσβασης, κατακερματισμός κωδικών με salting για προστασία από επιθέσεις rainbow αλλά και για την ικανοποίηση απαιτήσεων ιδιωτικότητας, καθώς και προστασία από CSRF & XSS επιθέσεις.
Απόδοση: Χρόνος παραγωγής διαγράμματος ≤ 8 δευτερόλεπτα.
Αξιοπιστία: Το σύστημα θα πρέπει να χειρίζεται αξιόπιστα τα σφάλματα που μπορεί να προκύψουν είτε από την πλευρά του χρήστη είτε από την έξοδο του LLM. Ειδικότερα, το μήνυμα σφάλματος που εμφανίζεται στον χρήστη πρέπει να είναι κατάλληλα διαμορφωμένο, περιεκτικό και κατανοητό, παρέχοντας επαρκείς πληροφορίες για τη φύση και τη θέση του προβλήματος, ώστε να διευκολύνει την κατανόηση και την επανυποβολή
Φορητότητα: Η εφαρμογή θα πρέπει να μπορεί να εκτελείται σε οποιοδήποτε σύστημα ή πλατφόρμα.

4.1.4 Αποτίμηση Βαθμού Ικανοποίησης των Προδιαγεγραμμένων Απαιτήσεων

Η αξιολόγηση της εφαρμογής έδειξε ότι ικανοποιεί σε μεγάλο βαθμό τις λειτουργικές (τόσο υποχρεωτικές όσο και προαιρετικές) και μη λειτουργικές απαιτήσεις που προσδιορίστηκαν κατά τη φάση της ανάλυσης. Όσον αφορά τις υποχρεωτικές λειτουργικές απαιτήσεις, η εφαρμογή υποστηρίζει πλήρως την παραγωγή UML διαγραμμάτων τύπου κλάσεων, ακολουθίας, περιπτώσεων χρήσης και δραστηριοτήτων, καθώς και την επιλογή τύπου διαγράμματος και LLM από τον χρήστη. Επιπλέον, προσφέρει δυνατότητα εξαγωγής διαγραμμάτων σε μορφές PNG, SVG και ASCII, όπως είχε προβλεφθεί.

Όσον αφορά τις προαιρετικές λειτουργικές απαιτήσεις, αυτές υλοποιήθηκαν πλήρως. Ειδικότερα, η δυνατότητα τροποποίησης του παραγόμενου διαγράμματος από τον χρήστη μέσω PlantUML editor ενσωματώθηκε επιτυχώς και λειτουργεί σε όλα τα σενάρια. Επιπλέον, όλες οι λειτουργίες διαχείρισης χρηστών έχουν υλοποιηθεί πλήρως και αξιολογήθηκαν ως πλήρως λειτουργικές.

Σε ό,τι αφορά τις μη λειτουργικές απαιτήσεις, η εφαρμογή παρουσιάζει πολύ καλή απόδοση ως προς τον χρόνο παραγωγής διαγράμματος, με τις περισσότερες περιπτώσεις να ολοκληρώνονται εντός 2 δευτερολέπτων. Η διεπαφή χρήστη είναι πλήρως αποκρίσιμη και προσαρμόζεται σε διαφορετικές συσκευές, ενώ έχουν εφαρμοστεί όλα τα βασικά μέτρα ασφάλειας (token-based authentication, RBAC, κατακερματισμός κωδικών, προστασία από XSS και CSRF επιθέσεις). Η φορητότητα καλύπτεται πλήρως, καθώς η εφαρμογή είναι υλοποιημένη σε Python/Django — τεχνολογίες ανεξάρτητες πλατφόρμας — και μπορεί να εκτελείται σε Windows, macOS ή Linux, αρκεί να υπάρχει εγκατεστημένη η Python και ένας web browser. Επιπλέον, η χρήση εικονικού (virtual) περιβάλλοντος (virtualenv) και η τεκμηριωμένη διαδικασία εγκατάστασης διασφαλίζουν την ομαλή αναπαραγωγή της εφαρμογής σε διαφορετικά συστήματα.

Συνολικά, το σύστημα ικανοποιεί σε μεγάλο βαθμό τις απαιτήσεις του αρχικού σχεδιασμού, παρέχοντας ένα ώριμο, σταθερό και χρήσιμο περιβάλλον για τη μοντελοποίηση UML διαγραμμάτων με τη βοήθεια LLMs.

4.2 Σχεδίαση Συστήματος

Η φάση της σχεδίασης αποτελεί κρίσιμο στάδιο στη διαδικασία ανάπτυξης, καθώς μεταφράζει τις λειτουργικές (και μη) απαιτήσεις του συστήματος σε **μορφοποιημένες οπτικές αναπαραστάσεις**. Τα διαγράμματα UML που δημιουργούνται κατά τη φάση αυτή βοηθούν στην κατανόηση της αρχιτεκτονικής, των ροών, των συστατικών και των αλληλεπιδράσεων του συστήματος. Όπως αναφέρθηκε και στο κεφάλαιο του υποβάθρου, τα διαγράμματα αυτά καλύπτουν τόσο την δομή όσο και την αναμενόμενη συμπεριφορά του συστήματος λογισμικού προς ανάπτυξη.

Στην παρούσα ενότητα παρουσιάζονται τα βασικά **σχεδιαστικά μοντέλα** που δημιουργήθηκαν για την αποτύπωση της δομής και συμπεριφοράς της εφαρμογής μας σε αντίστοιχες υπο-ενότητες.

4.2.1 Διάγραμμα Κλάσεων (Class Diagram)

Περιγραφή σε φυσική γλώσσα:

Το σύστημα περιλαμβάνει τρεις βασικές οντότητες που αναπαρίστανται ως κλάσεις:

1. User
Η κλάση User αναπαριστά τον χρήστη της εφαρμογής και περιλαμβάνει:

- *name*: το όνομα του χρήστη
- *email*: η διεύθυνση email
- *password*: ο κωδικός πρόσβασης (hashed)

- *role*: ο ρόλος (user ή admin)
- *age*: η ηλικία
- *securityQuestion*: η ερώτηση ασφαλείας
- *securityAnswer*: η απάντηση ασφαλείας

Σχέσεις:

- Ένας User έχει **πολλά Diagrams**.
- Ένας User σχετίζεται με **πολλά LLMs** (many-to-many) και μέσω αυτής της σχέσης αποθηκεύεται το accessToken ανά LLM.

2. Diagram

Η κλάση Diagram αναπαριστά ένα UML διάγραμμα και περιλαμβάνει:

- *diagramId*: μοναδικό αναγνωριστικό
- *type*: τύπος UML διαγράμματος (Class, Sequence, Use Case, Activity)
- *description*: περιγραφή σε φυσική γλώσσα
- *content*: κώδικας PlantUML
- *exportFormat*: μορφή εξαγωγής (PNG, SVG, ASCII)
- *size*: μέγεθος διαγράμματος

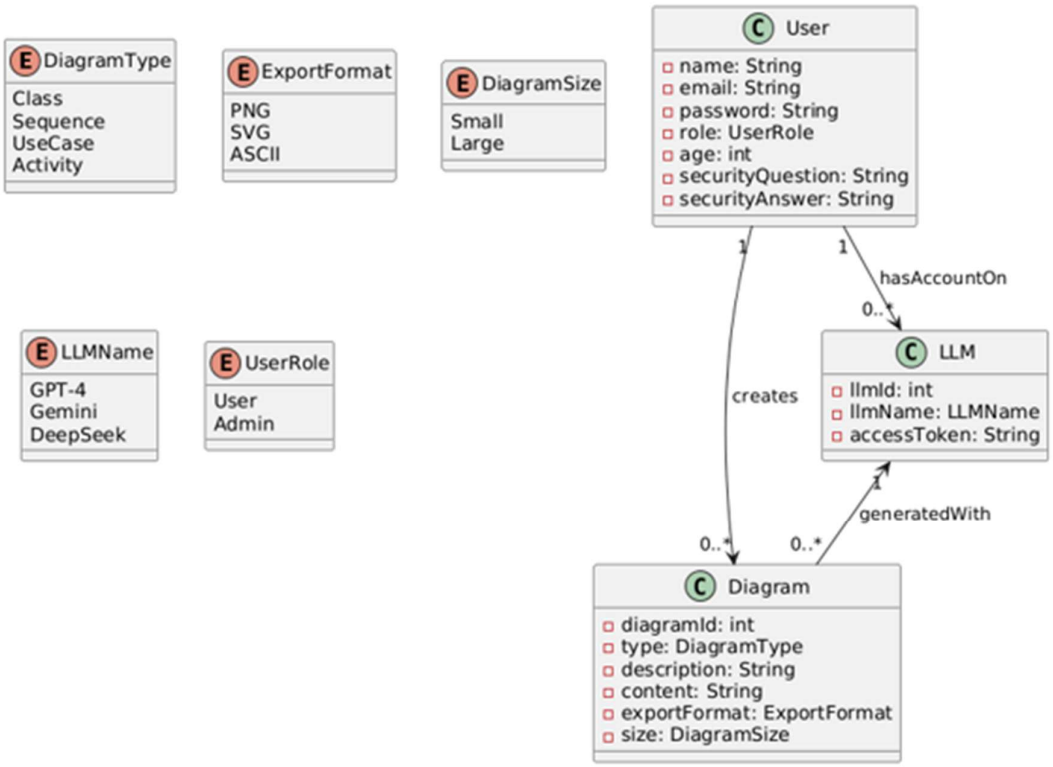
Σχέση: Κάθε Diagram σχετίζεται με ακριβώς **ένα LLM** .

3. LLM

Η κλάση LLM περιλαμβάνει:

- *llmId*: μοναδικό αναγνωριστικό
- *llmName*: όνομα LLM (π.χ. GPT-4, Gemini, DeepSeek)
- *accessToken*: το access token (αποθηκευμένο κρυπτογραφημένα)

Αποτελέσματα σχεδίασης - διάγραμμα κλάσεων:



Εικόνα 1. Διάγραμμα κλάσεων του συστήματος

4.2.2 Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram)

Περιγραφή σε φυσική γλώσσα:

Το σύστημα υποστηρίζει τρεις τύπους χρηστών:

- **Visitor (Επισκέπτης):** μη εγγεγραμμένος χρήστης, με δυνατότητα δημιουργίας UML διαγραμμάτων περιορισμένων τύπων. Δεν απαιτείται εγγραφή για την πρόσβαση στις βασικές λειτουργίες. Ο Visitor δεν μπορεί να επιλέξει LLM. Εκτός από την δημιουργία διαγράμματος, ο επισκέπτης μπορεί να κάνει και εγγραφή στο σύστημα.
- **RegisteredUser (Εγγεγραμμένος χρήστης):** έχει πρόσβαση σε όλες τις βασικές λειτουργίες της εφαρμογής, όπως επιλογή LLM, παροχή εισόδου (prompt), αποθήκευση, τροποποίηση και εξαγωγή διαγραμμάτων. Επιπλέον, μπορεί να διαχειρίζεται τον δικό του λογαριασμό.

- **Admin (Διαχειριστής):** αποτελεί εξειδίκευση του RegisteredUser, καθώς διαθέτει επιπλέον δικαιώματα διαχείρισης χρηστών, όπως αναζήτηση, επεξεργασία και έγκριση αιτήσεων διαγραφής λογαριασμού.

Οι Visitor και RegisteredUser εξειδικεύουν τη γενική οντότητα User. Ο User συνδέεται με τη βασική περίπτωση χρήσης Generate Diagram, η οποία μοντελοποιεί τη δημιουργία διαγράμματος UML.

Η Generate Diagram περιλαμβάνει τις εξής επιμέρους περιπτώσεις χρήσης:

- **Provide Input**: περιλαμβάνεται υποχρεωτικά (<<include>>) και αφορά την επιλογή τύπου διαγράμματος και την εισαγωγή περιγραφής σε φυσική γλώσσα.

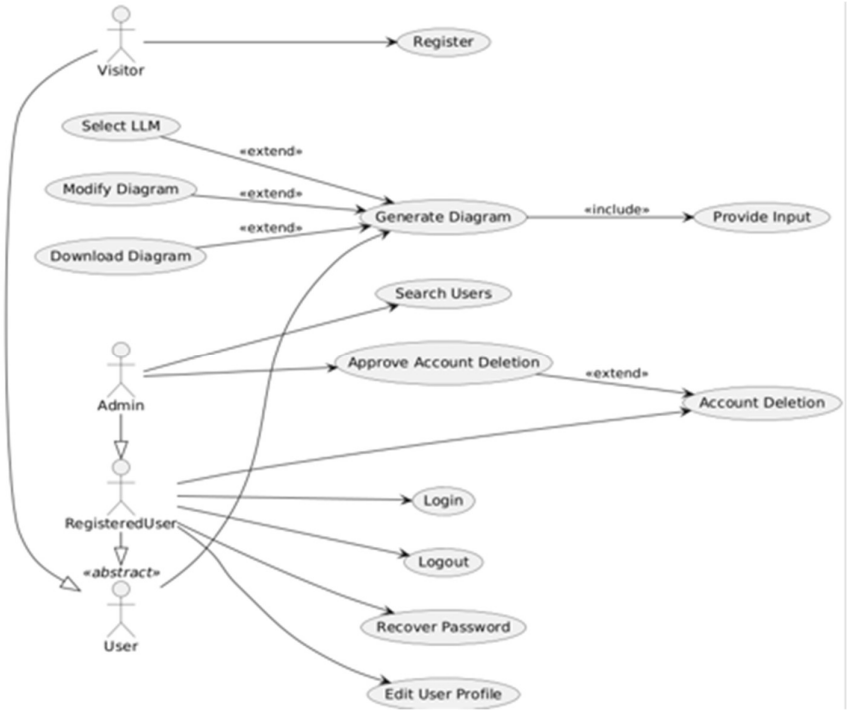
- **Select LLM**: δηλώνεται ως <<extend>> και ενεργοποιείται μόνο όταν ο χρήστης είναι εγγεγραμμένος.

- **Modify Diagram** και **Download Diagram**: αποτελούν επεκτάσεις (<<extend>>) της βασικής ροής της Generate Diagram, διαθέσιμες όταν έχει ήδη παραχθεί ένα διάγραμμα.

Ο Visitor μπορεί να εγγραφεί μέσω της λειτουργίας (περίπτωσης χρήσης) Register, ενώ ο RegisteredUser έχει δυνατότητες Login, Logout και Recover Password.

Η λειτουργία Account Deletion ενεργοποιείται από εγγεγραμμένο χρήστη, ενώ η Approve Account Deletion από Admin ενώ δηλώνεται ως <<extend>>, καθώς ενεργοποιείται μόνο αν έχει προηγηθεί αίτημα διαγραφής.

Αποτελέσματα σχεδίασης - διάγραμμα περιπτώσεων χρήσης:



Εικόνα 2. Διάγραμμα περιπτώσεων χρήσης συστήματος

4.2.3 Διάγραμμα Αρχιτεκτονικής (Architecture Diagram)

Περιγραφή σε φυσική γλώσσα:

Το μοντέλο αρχιτεκτονικής που παίρνει τη μορφή ενός μπλοκ διαγράμματος αναπαριστά τη λογική αρχιτεκτονική της εφαρμογής, διακρίνοντας εσωτερικά και εξωτερικά στοιχεία.

Η εφαρμογή αποτελείται από δύο βασικά υπο-συστήματα:

● Το WebApp (frontend), που λειτουργεί ως (γραφική) διεπαφή χρήστη.

● Το Backend, το οποίο περιλαμβάνει τα παρακάτω στοιχεία (components):

• **Main Controller**: Ενορχηστρώνει τα αιτήματα από το frontend και κατευθύνει τη ροή εκτέλεσης προς τα υπόλοιπα στοιχεία.

• **Security Module**: Διαχειρίζεται αυθεντικοποίηση (login/logout), έλεγχο ρόλων, ερωτήσεις ασφαλείας και tokens, καλύπτοντας πλήρως τις λειτουργίες ασφαλείας.

• **Diagram Logic Service**: Επεξεργάζεται την περιγραφή εισόδου του χρήστη, δημιουργεί το κατάλληλο prompt, το στέλνει στο εξωτερικό LLM API για επεξεργασία και λαμβάνει πίσω την σχετική απάντηση.

• **PlantUML Renderer**: Επικυρώνει τον παραγόμενο PlantUML κώδικα και τον μετατρέπει σε εικόνα.

• **Database**: Αποθηκεύει τις βασικές οντότητες της εφαρμογής, όπως χρήστες και διαγράμματα.

Το μόνο εξωτερικό σύστημα είναι το:

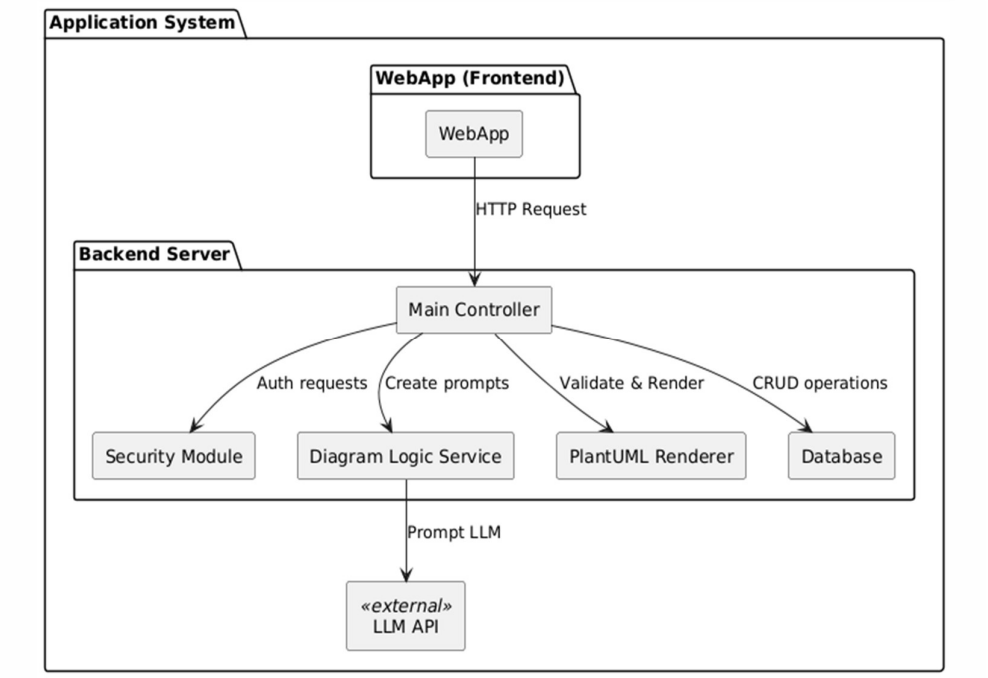
● **LLM API**: Εξωτερική RESTful υπηρεσία που δέχεται prompt από το Diagram Logic Service και επιστρέφει κώδικα PlantUML. Μπορεί να υλοποιείται με χρήση μοντέλων, όπως OpenAI GPT, Gemini ή DeepSeek.

Οι βασικές αλληλεπιδράσεις είναι οι εξής:

● Η WebApp επικοινωνεί μόνο με τον Main Controller μέσω HTTP αιτημάτων.

● Ο Main Controller διαχειρίζεται τη ροή των αιτημάτων προς τα επιμέρους modules/στοιχεία, χωρίς να αλληλεπιδρά απευθείας με το LLM API. Η αποστολή prompt στο LLM API γίνεται από το Diagram Logic Service.

Αποτελέσματα σχεδίασης - διάγραμμα αρχιτεκτονικής:



Εικόνα 3. Διάγραμμα συστατικών συστήματος

4.2.4 Διάγραμμα Ακολουθίας (Sequence Diagram)

Περιγραφή σε φυσική γλώσσα:

Το διάγραμμα ακολουθίας απεικονίζει τη ροή για την περίπτωση χρήσης Generate Diagram, καλύπτοντας επίσης τις υπο-περιπτώσεις (χρήσης) Provide Input, Select LLM, Modify Diagram και Download Diagram.

Ο χρήστης εισάγει μια περιγραφή προβλήματος μέσω της διεπαφής (Web UI) και επιλέγει τις παραμέτρους διαμόρφωσης (τύπο διαγράμματος, μοντέλο LLM, μέγεθος εξόδου). Η είσοδος αυτή μεταφέρεται στο backend (Main Controller) ως taskDescriptionAndConfiguration.

Το backend (Diagram Logic Service) δημιουργεί το prompt και το στέλνει στο επιλεγμένο LLM API. Πριν από την αποστολή, ελέγχεται (από τον Main Controller) αν

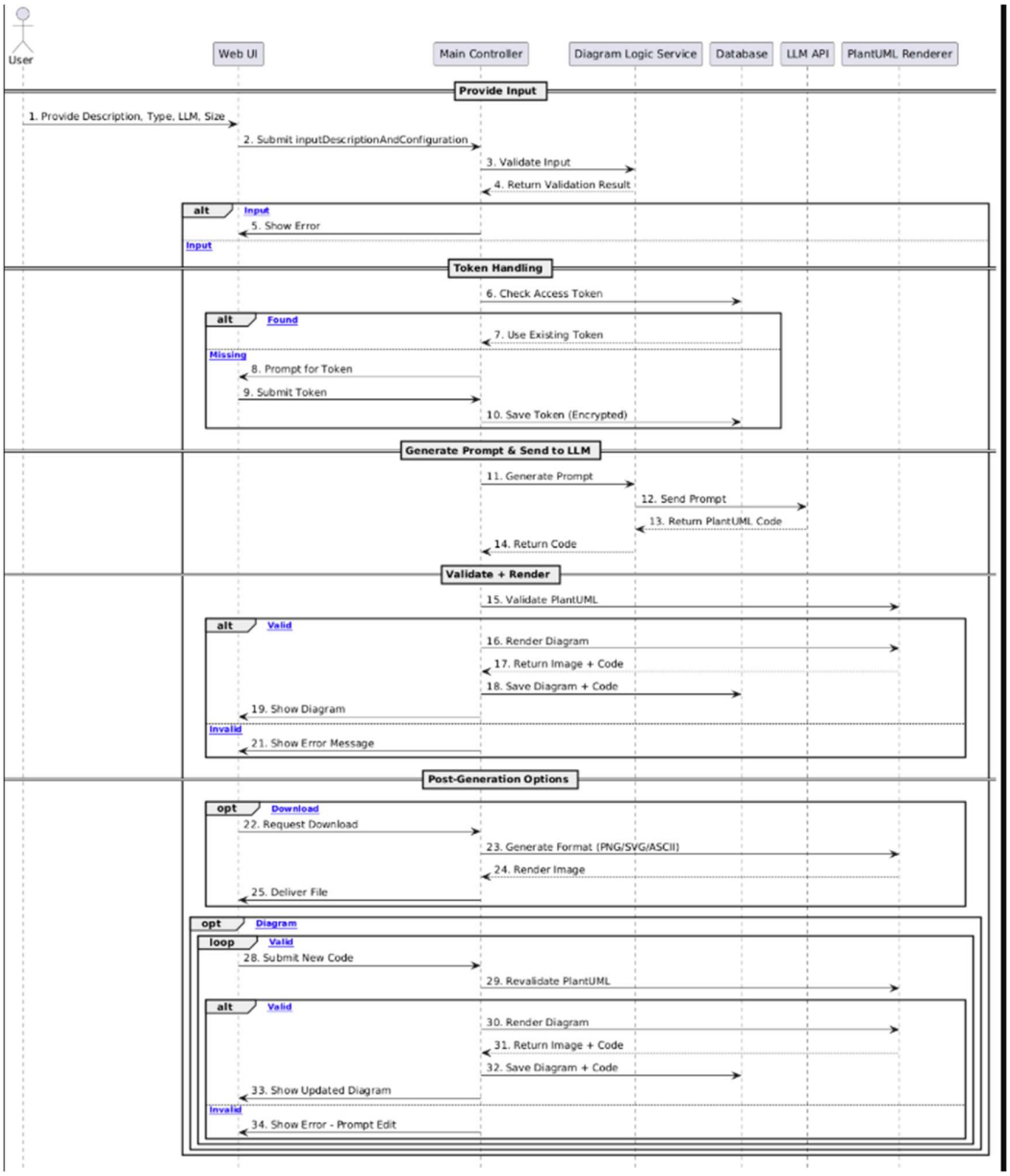
υπάρχει αποθηκευμένο access token για το συγκεκριμένο LLM. Το access token αποθηκεύεται μόνιμα στη βάση δεδομένων, κρυπτογραφημένο, για μελλοντική χρήση.

Το LLM API επιστρέφει ένα PlantUML αλφαριθμητικό (string), το οποίο αντιστοιχεί στο παραγόμενο μοντέλο διαγράμματος. Το backend (PlantUML Renderer) κάνει συντακτικό έλεγχο στο PlantUML string και, αν είναι έγκυρο, αποδίδει το διάγραμμα και αποθηκεύει το αποτέλεσμα στη βάση δεδομένων. Αν εντοπιστεί σφάλμα, επιστρέφεται μήνυμα σφάλματος στον χρήστη.

Ο χρήστης μπορεί στη συνέχεια να κατεβάσει την εικόνα του διαγράμματος (Download Diagram) (εφόσον αυτό είναι έγκυρο) ή να το τροποποιήσει (Modify Diagram). Αν τροποποιήσει τον PlantUML κώδικα, γίνεται νέος έλεγχος και rendering. Αν αλλάξει την περιγραφή, δημιουργείται νέο prompt και η διαδικασία ξεκινά από την αρχή.

Το διάγραμμα καλύπτει πλήρως τη λειτουργική ροή όλων των περιπτώσεων χρήσης που σχετίζονται με την παραγωγή UML διαγράμματος.

Αποτελέσματα σχεδίασης - διάγραμμα ακολουθίας:



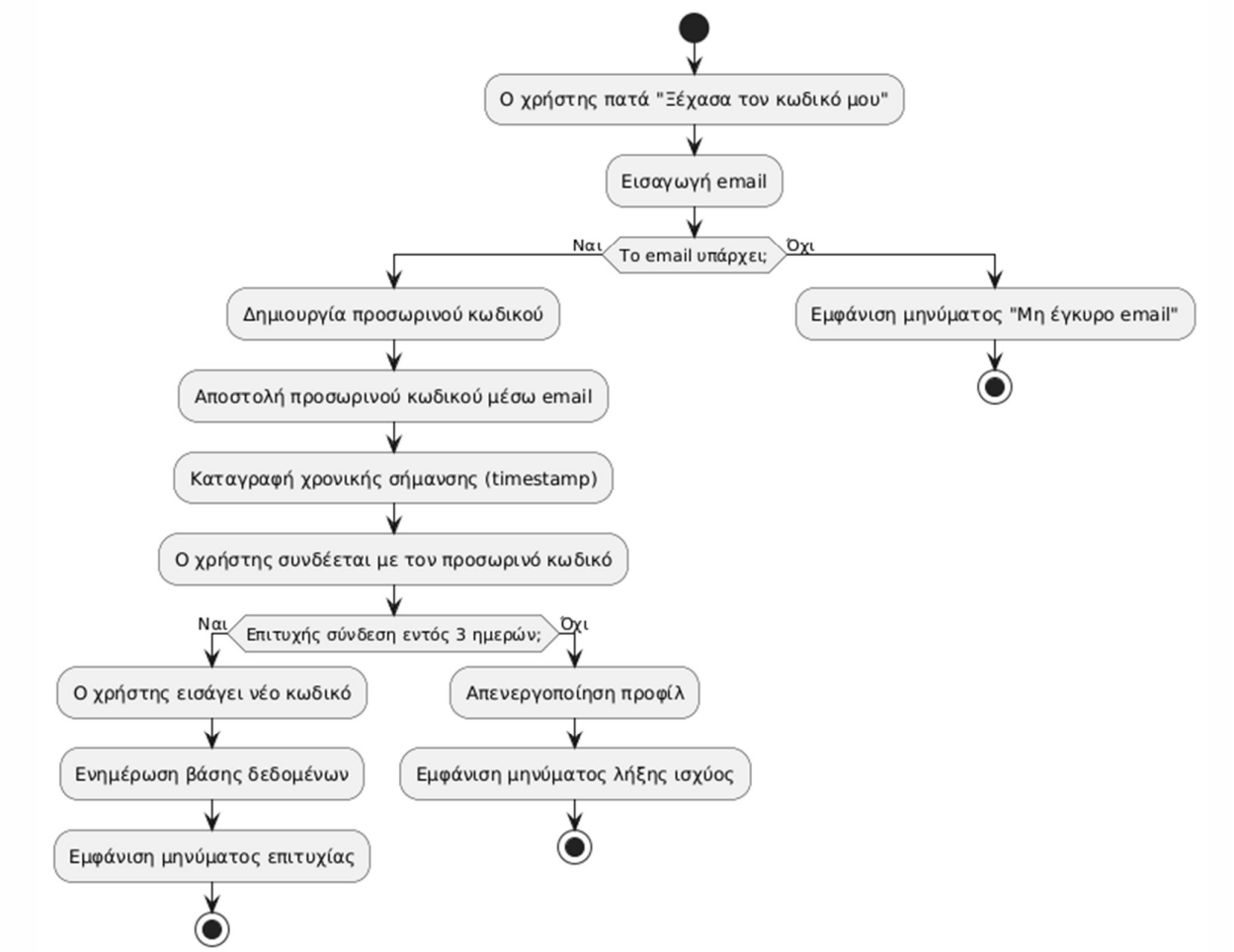
Εικόνα 4. Διάγραμμα ακολουθίας για τη δημιουργία UML διαγράμματος

4.2.5 Διαγράμματα Δραστηριοτήτων (Activity Diagram)

4.2.5.1 Ανάκτηση Κωδικού Χρήστη

Περιγραφή σε φυσική γλώσσα:

Η διαδικασία ανάκτησης (δείτε Εικόνα 5) ενεργοποιείται όταν ο χρήστης πατήσει "Ξέχασα τον κωδικό μου". Το σύστημα αποστέλλει **προσωρινό κωδικό στο email** του χρήστη, με ισχύ **3 ημερών**. Ο χρήστης οφείλει να συνδεθεί με αυτόν και να αλλάξει τον κωδικό του εντός του χρονικού ορίου. Αν παρέλθει το όριο χωρίς αλλαγή, το προφίλ απενεργοποιείται αυτόματα.

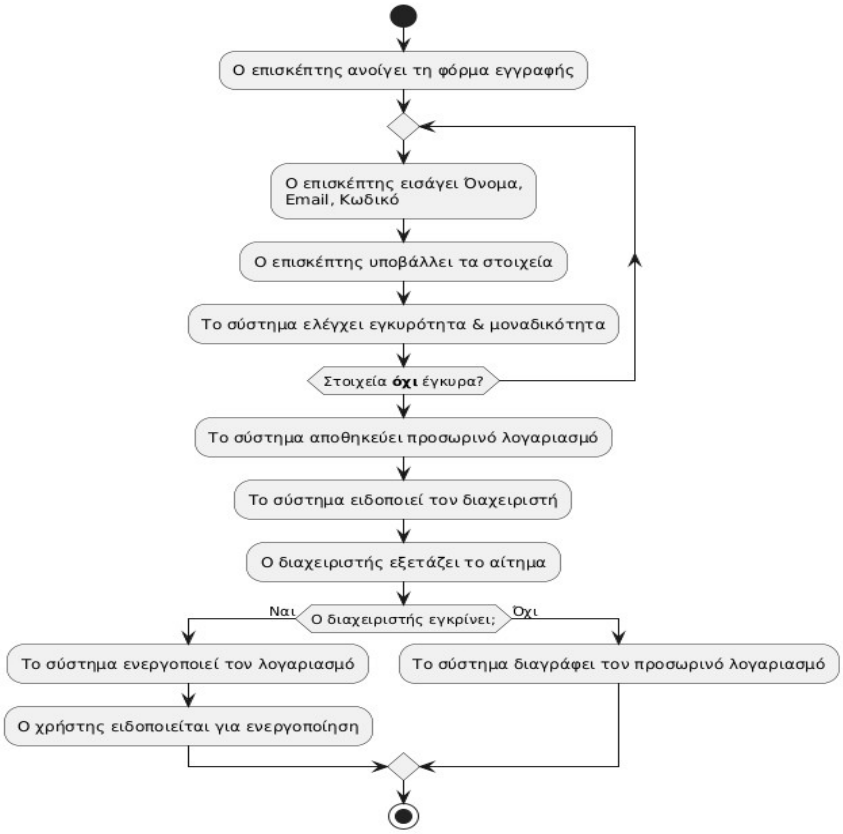


Εικόνα 5. Διάγραμμα δραστηριοτήτων για την ανάκτηση κωδικού χρήστη

4.2.5.2 Εγγραφή Χρήστη

Περιγραφή σε φυσική γλώσσα:

Το διάγραμμα δραστηριοτήτων (δείτε Εικόνα 6) απεικονίζει τη ροή για τη διαδικασία εγγραφής νέου χρήστη στην εφαρμογή. Ο επισκέπτης ανοίγει τη φόρμα εγγραφής και εισάγει τα απαραίτητα στοιχεία (όνομα χρήστη, email, κωδικό πρόσβασης). Το σύστημα ελέγχει την εγκυρότητα και τη μοναδικότητα των δεδομένων. Αν ο έλεγχος είναι επιτυχής, ο λογαριασμός αποθηκεύεται προσωρινά στη βάση δεδομένων. Στη συνέχεια, ο διαχειριστής (Admin) ελέγχει τα στοιχεία του χρήστη και εγκρίνει ή απορρίπτει την εγγραφή. Αν εγκριθεί, ο χρήστης μπορεί να εισέλθει κανονικά στο σύστημα. Αν απορριφθεί, ο λογαριασμός διαγράφεται

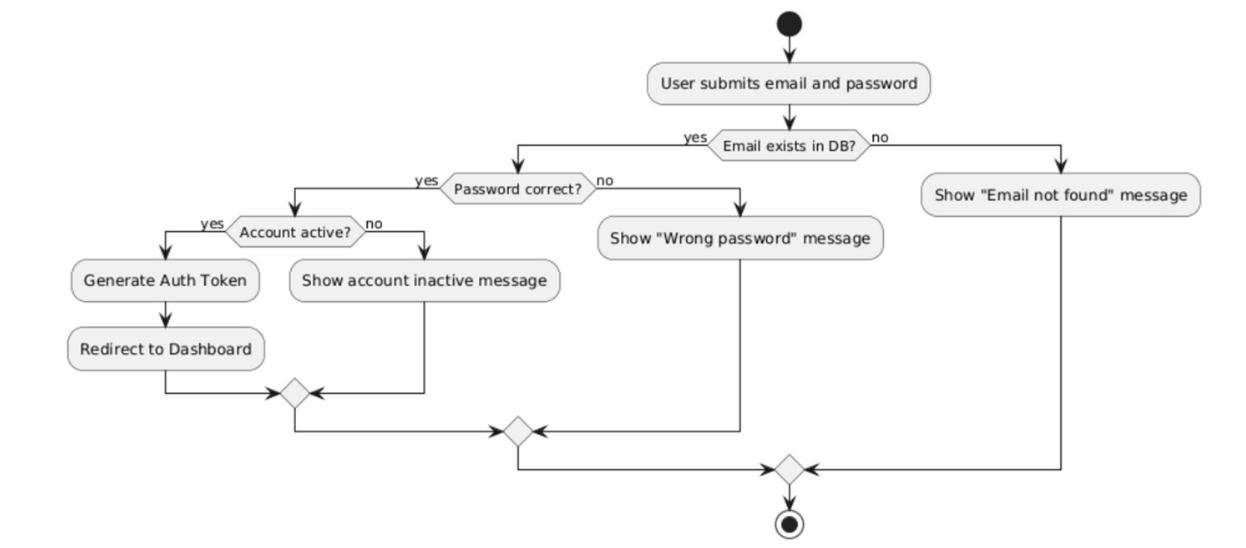


Εικόνα 6. Διάγραμμα δραστηριοτήτων για την εγγραφή χρήστη

4.2.5.3 Login χρήστη (Αυθεντικοποίηση)

Περιγραφή σε φυσική γλώσσα:

Η δραστηριότητα αυθεντικοποίησης (login) (δείτε Εικόνα 7) ενεργοποιείται όταν ο χρήστης εισάγει το email και τον κωδικό πρόσβασης. Το σύστημα ελέγχει αν το email αντιστοιχεί σε υπάρχοντα λογαριασμό και αν ο κωδικός είναι σωστός. Εάν ναι, τότε εφόσον ο λογαριασμός του χρήστη είναι ενεργοποιημένος, δημιουργείται και αποστέλλεται ένα authentication token. Αν κάποιο από τα στοιχεία είναι λάθος ή ο λογαριασμός του χρήστη είναι απενεργοποιημένος, εμφανίζεται σχετικό μήνυμα σφάλματος. Ο λογαριασμός του χρήστη ενδέχεται να είναι απενεργοποιημένος εφόσον ακόμη δεν έχει εγκριθεί η εγγραφή του από τον διαχειριστή.



Εικόνα 7. Διάγραμμα Login χρήστη (Αυθεντικοποίηση)

4.3 Υλοποίηση

Η εφαρμογή αναπτύχθηκε με στόχο την επίτευξη αρθρωτής, επεκτάσιμης και ασφαλούς αρχιτεκτονικής. Βασίστηκε στο πλαίσιο Django (Python) (ειδικότερα όσον αφορά το Backend), το οποίο προσφέρει υψηλού επιπέδου λειτουργικότητες για την κατασκευή RESTful εφαρμογών με ενσωματωμένη υποστήριξη διαχείρισης χρηστών, ασφάλειας και δρομολόγησης (routing). Το Frontend υλοποιήθηκε με HTML, CSS και JavaScript, αξιοποιώντας το πλαίσιο Bootstrap για την υλοποίηση ενός αποκρίσιμου (responsive) σχεδιασμού της διεπαφής χρήστη, ώστε αυτή να εμφανίζεται και να ανταποκρίνεται ορθά σε διαφορετικές συσκευές και αναλύσεις οθόνης.

4.3.1 Τεχνολογίες & Βιβλιοθήκες

Ο παρακάτω πίνακας προσδιορίζει τις τεχνολογίες και βιβλιοθήκες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής ιστού.

Πίνακας 3. Τεχνολογίες & Βιβλιοθήκες

Τεχνολογία	Τύπος Τεχνολογίας	Ρόλος
Python 3.x	Γλώσσα προγραμματισμού	Python 3.x ως κύρια γλώσσα υλοποίησης / προγραμματισμού του backend και βοηθητικών scripts (π.χ. migrations, API clients, δοκιμές)

Django	Πλαίσιο (framework)	Πλαίσιο ανάπτυξης Backend εφαρμογής
Django REST Framework	Βιβλιοθήκη Django	Δημιουργία RESTful τελικών σημείων (endpoints) για επικοινωνία με το frontend
Requests	Βιβλιοθήκη Python	Επικοινωνία με εξωτερικά APIs, ιδίως για αποστολή prompt σε LLMs (δηλ. επικοινωνία με LLM APIs)
PlantUML	Εργαλείο/Renderer	Μετατροπή PlantUML script σε εικόνα (SVG/PNG), καθώς και έλεγχος συντακτικής εγκυρότητας του script
SQLite / PostgreSQL	Συστήματα βάσεων δεδομένων	Το SQLite χρησιμοποιείται για δοκιμές, το PostgreSQL για παραγωγικό περιβάλλον
Bootstrap	Βιβλιοθήκη CSS	Styling και αποκρίσιμος σχεδιασμός του UI
JavaScript	Γλώσσα προγραμματισμού	Δυναμική λειτουργικότητα του Frontend (web UI)
pytest / unittest	Εργαλεία δοκιμών	Αυτοματοποιημένες δοκιμές στο Backend

4.3.2 Δομή Κώδικα

Η δομή του έργου ακολουθεί την τυπική αρχιτεκτονική ενός Django project (έργου), με επιπλέον φακέλους (π.χ. management, templatetags, emails, swaggers) και αρχεία (π.χ. test_openai.py) που εξυπηρετούν τη λειτουργικότητα παραγωγής διαγραμμάτων και την επικοινωνία με LLM APIs.

umlgenerator/

├── diagram_app/ — Κύρια Django εφαρμογή που περιλαμβάνει τη λογική της εφαρμογής, τα μοντέλα, τις φόρμες, τις προβολές (views) και τα APIs.

│ ├── management/ — Περιέχει custom Django commands, δηλαδή εντολές που επεκτείνουν τη γραμμή εντολών του Django (αρχεία .py).

│ ├── migrations/ — Περιέχει αρχεία Python που καταγράφουν τις αλλαγές στα μοντέλα για την ενημέρωση της βάσης δεδομένων (migrations).

│ ├── templates/ — Περιέχει τα HTML templates που χρησιμοποιούνται για την απόδοση των σελίδων στο frontend, συνδυάζοντας δυναμικά δεδομένα (μέσω του Django context) με ενσωματωμένο στατικό περιεχόμενο. Ο CSS και JavaScript κώδικας περιλαμβάνεται απευθείας εντός των αρχείων HTML. Ενδεικτικά templates: home.html, delete_account.html, user_profile.html, update_profile.html, password_reset.html κ.ά.

│ ├── templatetags/ — Περιέχει custom φίλτρα Django που χρησιμοποιούνται μέσα στα templates για ειδικές λειτουργίες (αρχεία .py).

│ ├── __init__.py — Δηλώνει τον φάκελο ως Python module.

│ ├── admin.py — Περιέχει την εγγραφή και παραμετροποίηση μοντέλων ώστε να είναι προσβάσιμα και διαχειρίσιμα από το Django admin interface.

│ ├── apps.py — Ρυθμίσεις της εφαρμογής (application configuration).

│ ├── forms.py — Ορισμός Django φορμών για διαχείριση εισαγωγής δεδομένων από τον χρήστη.

│ ├── models.py — Ορισμός των ORM μοντέλων (π.χ. User, Diagram, Project) που χαρτογραφούν τη βάση δεδομένων.

│ ├── serializers.py — DRF serializers για μετατροπή μοντέλων σε JSON και αντίστροφα, υποστηρίζοντας τα REST API endpoints.

│ ├── signals.py — Ορισμός Django signals για αυτόματες ενέργειες (π.χ. logging) κατά την τροποποίηση αντικειμένων.

│ ├── tests.py — Περιέχει μονάδες ελέγχου (unit tests) για τον έλεγχο της λειτουργικότητας.

│ ├── urls.py — Ρυθμίζει τα URLs της εφαρμογής (routing).

│ ├── utils.py — Βοηθητικές συναρτήσεις για την εφαρμογή (π.χ. rendering, υποστήριξη διεργασιών).

│ └── views.py — Περιέχει τις βασικές προβολές (views) που χειρίζονται τα αιτήματα των χρηστών και καθορίζουν τις αποκρίσεις. Στη περίπτωση REST APIs, περιλαμβάνει και τα API endpoints.

│

├── emails/ — Περιέχει utilities για την αποστολή και διαχείριση email λειτουργιών (αρχεία Python).

│

├── uml_generator/ — Περιέχει τις κεντρικές ρυθμίσεις του Django project, συμπεριλαμβανομένων των παραμέτρων εκκίνησης.

│ ├── __init__.py — Δηλώνει τον φάκελο ως Python module.

│ ├── asgi.py — Ρυθμίσεις ASGI για ασύγχρονη εκτέλεση εφαρμογών Django.

│ ├── settings.py — Κεντρικές ρυθμίσεις της εφαρμογής, όπως σύνδεση σε βάση, middleware, εγκατεστημένα apps.

│ ├── urls.py — Κεντρικό routing του project που δρομολογεί URLs προς τις κατάλληλες εφαρμογές.

│ └── wsgi.py — Ρυθμίσεις WSGI για εκτέλεση σε παραγωγικά περιβάλλοντα με servers όπως Gunicorn/uWSGI.

│

├── db.sqlite3 — SQLite βάση δεδομένων για την αποθήκευση των δεδομένων του συστήματος.

├── manage.py — Entry point για εκτέλεση Django εντολών από τη γραμμή εντολών (π.χ. runserver, migrate).

├── requirements.txt — Λίστα με τις Python βιβλιοθήκες/εξαρτήσεις που χρειάζεται το project για να τρέξει.

├── project_structure.txt — Τεκμηρίωση της δομής του έργου, για αναφορά.

├── test_openai.py — Αρχείο δοκιμών για τον έλεγχο επικοινωνίας με το OpenAI API.

├── swaggers/ — Περιέχει αρχεία τεκμηρίωσης API σε μορφή Swagger/OpenAPI για περιγραφή και

4.3.3 Πληροφοριακές Κλάσεις

Η εφαρμογή βασίζεται σε ένα σύνολο πληροφοριακών κλάσεων, οι οποίες καταγράφουν βασική πληροφορία που είναι απαραίτητη για την υλοποίηση των βασικών λειτουργιών του συστήματος. Επομένως, οι κλάσεις αυτές, οι οποίες αποτυπώνονται και στο διάγραμμα κλάσεων, αποτελούν τον πυρήνα της επιχειρησιακής λογικής.

User

Αναπαριστά τον χρήστη του συστήματος και βασίζεται στο ενσωματωμένο μοντέλο User του Django. Η λειτουργικότητα επεκτείνεται μέσω της κλάσης *UserProfile*, η οποία σχετίζεται one-to-one με τον χρήστη (μοντέλο User) και περιλαμβάνει:

- *age*: Η ηλικία του χρήστη
- *security_question*: Η ερώτηση ασφαλείας για ανάκτηση κωδικού
- *security_answer*: Η απάντηση στην ερώτηση ασφαλείας
- *password_reset_requested_at*: Χρονοσφραγίδα αιτήματος επαναφοράς κωδικού

Ο χρήστης συσχετίζεται με διαγράμματα (Diagram), ενώ διαθέτει δυνατότητα αποθήκευσης πολλαπλών LLM tokens μέσω της κλάσης LLM.

Diagram

Αναπαριστά ένα UML διάγραμμα που δημιουργείται από τον χρήστη. Περιλαμβάνει:

- *diagram_id*: Το μοναδικό αναγνωριστικό του διαγράμματος
- *type*: Ο τύπος του διαγράμματος (π.χ. class, sequence), ως enum DiagramType
- *description*: Η περιγραφή του διαγράμματος από τον χρήστη σε φυσική γλώσσα
- *content*: Ο παραγόμενος κώδικας σε PlantUML
- *export_format*: Η μορφή εξαγωγής (PNG, SVG, ASCII) του διαγράμματος
- *size*: Το μέγεθος/λεπτομέρεια διαγράμματος, ως enum DiagramSize
- *llm: Ο πάροχος LLM που χρησιμοποιήθηκε, ως enum LLMName*

LLM

Αποθηκεύει πληροφορίες για τις υπηρεσίες LLM που χρησιμοποιεί κάθε χρήστης:

- *llm_id*: Αναγνωριστικό της υπηρεσίας
- *llm_name*: Το όνομα του μοντέλου (π.χ. GPT-4, Gemini)
- *access_token*: Το αντίστοιχο token, αποθηκευμένο με ασφάλεια

Κάθε χρήστης μπορεί να διαθέτει πολλαπλά tokens (π.χ. για GPT, Gemini) για διαφορετικά LLMs, γεγονός που επιτρέπει ευελιξία στη χρήση διαφορετικών μοντέλων.

UMLInput

Η κλάση UMLInput αναπαριστά το αίτημα που στέλνει ο χρήστης για παραγωγή διαγράμματος. Καταχωρείται στη βάση δεδομένων, ώστε να διατηρείται ιστορικό αιτημάτων, να είναι δυνατή η αναπαραγωγή του διαγράμματος και να υποστηρίζεται η αξιολόγηση της ποιότητας εισόδου/εξόδου.

Η κλάση συσχετίζεται:

- με τον χρήστη που υπέβαλε το αίτημα (User)

- και με το παραγόμενο διάγραμμα (Diagram), μέσω σχέσης 1:1

Περιλαμβάνει:

- user_input: Το κείμενο σε φυσική γλώσσα που πληκτρολογεί ο χρήστης

- diagram_type: Ο τύπος UML διαγράμματος (π.χ. Class, Sequence)
- llm_choice: Το μοντέλο LLM που θα χρησιμοποιηθεί (π.χ. GPT, Gemini)
- diagram_size: Ρύθμιση πολυπλοκότητας του διαγράμματος (π.χ. Small, Large)
- api_key: Το κλειδί πρόσβασης στο LLM που χρησιμοποίησε ο χρήστης

DeleteAccountRequest

Η κλάση DeleteAccountRequest αναπαριστά ένα αίτημα χρήστη για διαγραφή του λογαριασμού του. Κάθε αίτημα σχετίζεται με έναν συγκεκριμένο χρήστη του συστήματος.

- submitted_at: Χρονική σφραγίδα (timestamp) που δηλώνει πότε υποβλήθηκε το αίτημα.
- is_approved: Λογική τιμή (Boolean) που δείχνει αν το αίτημα εγκρίθηκε (True) ή απορρίφθηκε (False) από τον διαχειριστή.
- user: Αναφορά στον χρήστη (User) που υπέβαλε το αίτημα (σχέση 1:N ή 1:1, ανάλογα με την υλοποίηση).

4.3.4 Υλοποίηση Επιχειρησιακής Λογικής

Η εφαρμογή ακολουθεί αρχιτεκτονική προσανατολισμένη σε **διαχωρισμό ευθυνών (Separation of Concerns)**, με στόχο την ευελιξία, την επαναχρησιμοποίηση και την ευκολία στη συντήρηση. Αν και η βασική λογική υλοποιείται σε Django views και μοντέλα, υιοθετείται λογικός διαχωρισμός ανάμεσα σε τρία βασικά επίπεδα:

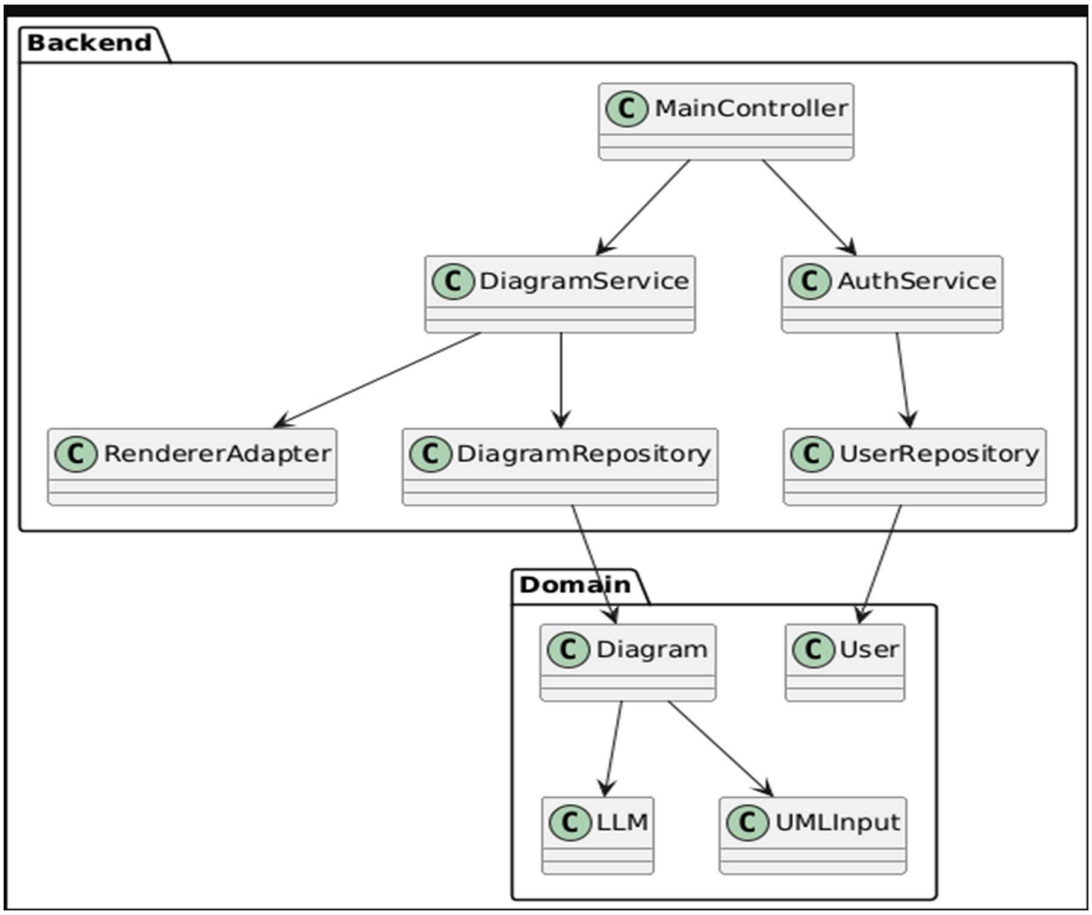
- **Controllers (views)**: Χειρίζονται τα αιτήματα από το frontend και κατευθύνουν τη ροή της εκτέλεσης.
- **Services**: Περιέχουν την επιχειρησιακή λογική του συστήματος, όπως η δημιουργία prompt, ο χειρισμός tokens και η επαλήθευση της εισόδου.
- **Repositories**: Διαχειρίζονται την πρόσβαση στα δεδομένα, λειτουργώντας ως ενδιάμεσο στρώμα μεταξύ του συστήματος και της βάσης δεδομένων. Αν και στο Django τα Repositories **δεν υλοποιούνται ως ξεχωριστές κλάσεις**, η χρήση του **Django ORM (Model.objects)** καλύπτει αυτόν τον ρόλο, προσφέροντας αφαιρετική διεπαφή για CRUD λειτουργίες. Έτσι, ο προγραμματιστής μπορεί να χειρίζεται τα δεδομένα με υψηλού επιπέδου εντολές, χωρίς να γράφει απευθείας SQL.

Σε πιο πολύπλοκα έργα Django, μπορεί να υλοποιηθούν και **ρητά repositories** ως ανεξάρτητες Python κλάσεις, με στόχο τον περαιτέρω διαχωρισμό λογικής, την ευκολότερη δοκιμή μονάδων (unit testing) και την επαναχρησιμοποίηση ερωτημάτων.

Υλοποιητικές Κλάσεις – Αρχιτεκτονική Backend

Στο παρακάτω διάγραμμα αποτυπώνεται ο διαχωρισμός της λογικής σε ελεγκτές (Controllers), υπηρεσίες (Services) και αποθετήρια (Repositories), καθώς και η

διασύνδεσή τους με τις βασικές domain οντότητες. Το διάγραμμα δείχνει τη ροή μεταξύ των υλοποιητικών επιπέδων και τη σχέση τους με τις πληροφοριακές κλάσεις.



Εικόνα 8. Υλοποιητικές Κλάσεις – Αρχιτεκτονική Backend

Ενδεικτικές Κλάσεις:

DiagramService - Υλοποιεί τις λειτουργίες παραγωγής UML διαγραμμάτων.

```
class DiagramService:
    def generate_prompt(self, user_input: UMLInput) -> str:
        # δημιουργία κατάλληλου prompt για LLM
        ...

    def save_diagram(self, diagram: Diagram):
        # αποθήκευση του παραγόμενου διαγράμματος
        ...
```

AuthService - Υπεύθυνη για αυθεντικοποίηση, διαχείριση tokens και έλεγχο πρόσβασης.

```
class AuthService:
    def authenticate_user(self, email, password) -> bool: ...
    def generate_token(self, user) -> str: ...
    def validate_token(self, token) -> bool: ...
```

DiagramController - Δέχεται αιτήματα από το frontend και χρησιμοποιεί τα services.

```
class DiagramController:
    def create_diagram(self, request):
        input_data = extract_input(request)
        prompt = DiagramService().generate_prompt(input_data)
        code = LLMClient().send_prompt(prompt)
        diagram = Renderer().render(code)
        DiagramService().save_diagram(diagram)
        return JsonResponse(diagram)
```

LLMClient - Περιέχει τη λογική επικοινωνίας με εξωτερικά LLM APIs.

```
class LLMClient:
    def send_prompt(self, prompt: str) -> str:
        # αποστολή prompt και λήψη απάντησης
        ...
```

4.3.5 Παραγωγή Prompt από Είσοδο Χρήστη

Κατά τη φάση δημιουργίας ενός νέου διαγράμματος, ο εγγεγραμμένος χρήστης καλείται να συμπληρώσει μία φόρμα με τα εξής στοιχεία:

- **Τύπος διαγράμματος** (Class, Sequence, Use Case, Activity)
- **Μέγεθος εξόδου** (Small ή Large)
- **Περιγραφή σε φυσική γλώσσα** του σεναρίου ή της λειτουργικότητας

Αυτά τα στοιχεία μετατρέπονται δυναμικά σε **prompt**, το οποίο αποστέλλεται στο επιλεγμένο LLM για την παραγωγή του αντίστοιχου διαγράμματος σε PlantUML. Ο μηχανισμός παραγωγής του prompt ακολουθεί **προκαθορισμένο πρότυπο (prompt template)** που βασίζεται σε **δύο βασικά μοτίβα**:

- **Role pattern:** Καθορίζει τον ρόλο που θα αναλάβει το LLM (π.χ. «Εργάσου σαν ειδικός στην δημιουργία UML διαγραμμάτων…»). Με αυτόν τον τρόπο, το μοντέλο καθοδηγείται να σκεφτεί και να παράγει έξοδο ως ειδικός τομέα (domain expert).
- **Output pattern:** Καθορίζει την επιθυμητή μορφή εξόδου, δηλαδή ότι το τελικό αποτέλεσμα πρέπει να είναι **αποκλειστικά έγκυρος PlantUML κώδικας**, χωρίς πρόσθετο κείμενο ή σχόλια.

Η συνδυαστική χρήση των δύο μοτίβων **βελτιώνει την ποιότητα** του παραγόμενου διαγράμματος, διασφαλίζοντας ότι το LLM κατανοεί επακριβώς **τι πρέπει να παράξει** και σε ποια μορφή.

Ενδεικτικό παράδειγμα μετατροπής σε prompt:

```
prompt = _(f"""

    Εργάσου σαν ειδικός στην δημιουργία διαγραμμάτων UML και
δημιούργησε ένα {diagram_type} διάγραμμα

    που θα είναι { 'μεγάλο' if diagram_size == 'large' else 'μικρό'
} σε μορφή PlantUML βασισμένο στην εξής περιγραφή:



    {user_input}



    Δώσε μόνο τον κώδικα PlantUML χωρίς εξηγήσεις.

    """
```

Κεφάλαιο 5 – Εγκατάσταση & Επίδειξη Συστήματος

5.1 Εγκατάσταση

Η εφαρμογή ιστού διατίθεται δημόσια στο διαδίκτυο μέσω της υπηρεσίας Render. Αυτό επιτρέπει σε κάθε ενδιαφερόμενο χρήστη να την επισκεφθεί και να την χρησιμοποιήσει άμεσα μέσω ενός απλού φυλλομετρητή (web browser), χωρίς να απαιτείται προηγούμενη εγκατάσταση ή ρύθμιση τοπικού περιβάλλοντος. Η live έκδοση της εφαρμογής είναι προσβάσιμη στον σύνδεσμο: https://umlai.onrender.com.

Παράλληλα, για σκοπούς τεκμηρίωσης και αξιολόγησης, είναι σημαντικό να περιγραφεί αναλυτικά η διαδικασία τοπικής εγκατάστασης, ώστε η εφαρμογή να μπορεί να αναπαραχθεί ή και να επεκταθεί από οποιονδήποτε διαθέτει πρόσβαση στον πηγαίο κώδικά της. Το αποθετήριο βρίσκεται στο GitHub, στο σύνδεσμο https://github.com/Billbourg/UMLaI, και είναι ιδιωτικό, ωστόσο μπορεί να δοθεί πρόσβαση σε αξιολογητές ή επιβλέποντες κατόπιν αιτήματος.

Προϋποθέσεις για Επιτυχή Εγκατάσταση

Πριν ξεκινήσει η εγκατάσταση της εφαρμογής σε τοπικό περιβάλλον, είναι απαραίτητο να πληρούνται οι εξής προϋποθέσεις:

- *Python*: Έκδοση 3.10 ή μεταγενέστερη (προτείνεται 3.11) — Python Downloads
- *Λειτουργικό σύστημα*: Windows 10+, macOS ή διανομή Linux (Ubuntu 20.04+)
- *Διαθέσιμος χώρος στο δίσκο*: ≥ 500 MB για το περιβάλλον και τις εξαρτήσεις
- *Εγκατεστημένο Git*: Για την κλωνοποίηση του αποθετηρίου — Git Downloads

Για να εκτελέσει κάποιος την εφαρμογή τοπικά, απαιτείται αρχικά η κλωνοποίηση του αποθετηρίου στον υπολογιστή του με την εντολή:

git clone <repository_url> όπου <repository_url> είναι ο προαναφερόμενος σύνδεσμος στο Github αποθετήριο του πηγαίου κώδικα της εφαρμογής

Στη συνέχεια, γίνεται μετάβαση στον ριζικό φάκελο του έργου:

cd UMLaI-main

Ακολουθεί η δημιουργία εικονικού περιβάλλοντος Python και η ενεργοποίησή του σε Linux/Mac:

bash

python3 -m venv env

source env/bin/activate

ή σε Windows:

bash

```
python -m venv env

env\Scripts\activate
```

Αφού ενεργοποιηθεί το εικονικό περιβάλλον, γίνεται η εγκατάσταση των απαιτούμενων εξαρτήσεων μέσω της εντολής:

bash

```
pip install -r requirements.txt
```

Η εντολή αυτή χρησιμοποιεί το αρχείο requirements.txt, το οποίο περιέχει καταγεγραμμένες όλες τις βιβλιοθήκες και εκδόσεις που απαιτούνται για την εκτέλεση της εφαρμογής. Με αυτόν τον τρόπο, εξασφαλίζεται ότι το τοπικό περιβάλλον θα έχει όλες τις απαραίτητες εξαρτήσεις εγκατεστημένες.

Για αναλυτικές οδηγίες σχετικά με τη δημιουργία και χρήση εικονικού περιβάλλοντος Python, μπορείτε να ανατρέξετε στην επίσημη τεκμηρίωση της βιβλιοθήκης venv: https://docs.python.org/3/library/venv.html

Η βάση δεδομένων, που είναι προεπιλεγμένα υλοποιημένη με **SQLite**, επιτρέπει την άμεση εκτέλεση της εφαρμογής τοπικά χωρίς περαιτέρω ρυθμίσεις, καθώς δημιουργείται αυτόματα με την εκτέλεση των ακόλουθων εντολών migration. Αυτό είναι ιδιαίτερα χρήσιμο για σκοπούς δοκιμών, αξιολόγησης και διάταξης σε περιβάλλοντα ανάπτυξης (development).

bash

```
python manage.py makemigrations

python manage.py migrate
```

Για **παραγωγική χρήση**, συνιστάται η διασύνδεση με **PostgreSQL**, ώστε να εξασφαλίζεται μεγαλύτερη αξιοπιστία και επεκτασιμότητα της βάσης δεδομένων. Σε αυτή την περίπτωση, ο χρήστης θα πρέπει:

- να έχει **προεγκαταστήσει** και να εκτελεί **PostgreSQL** σε server ή τοπικά,
- να δημιουργήσει την απαιτούμενη βάση δεδομένων και λογαριασμό χρήστη με τα κατάλληλα δικαιώματα,
- και να ενημερώσει κατάλληλα τις ρυθμίσεις της εφαρμογής (αρχείο settings.py του Django), τροποποιώντας την παράμετρο DATABASES ώστε να συνδέεται στο PostgreSQL instance αντί για την SQLite.

Καθώς η εφαρμογή διαθέτει σύστημα αυθεντικοποίησης, είναι αναγκαία η δημιουργία τουλάχιστον ενός διαχειριστή (superuser). Η δημιουργία του διαχειριστή γίνεται μέσω της εντολής:

bash

```
python manage.py createsuperuser
```

Κατά την εκτέλεση αυτής της εντολής ορίζονται όνομα χρήστη, email και κωδικός πρόσβασης.

Ακολουθώντας αυτά τα βήματα, ο χρήστης μπορεί να εγκαταστήσει και ρυθμίσει την εφαρμογή πλήρως τοπικά, χωρίς να εξαρτάται από την online εκδοχή της ή από εξωτερικές υπηρεσίες φιλοξενίας.

Η εφαρμογή μπορεί πλέον να ξεκινήσει τοπικά με την εντολή:

bash

```
python manage.py runserver
```

και καθίσταται προσβάσιμη τοπικά στη διεύθυνση http://127.0.0.1:8000, όπου ο χρήστης μπορεί να τη δοκιμάσει πλήρως, όπως και στην online έκδοση.

Η επιλογή της φιλοξενίας της εφαρμογής στο Render προσφέρει σημαντικά πλεονεκτήματα: εξαλείφει την ανάγκη εγκατάστασης από τον τελικό χρήστη, επιτρέπει αυτόματες ενημερώσεις μέσω GitHub (CI/CD), διευκολύνει την παρουσίαση και επίδειξη του συστήματος και επιτρέπει την κλιμάκωσή του σε περίπτωση ευρύτερης χρήσης. Η δημόσια διάθεση της εφαρμογής ενισχύει τη φορητότητα και επεκτασιμότητά της, ενώ η τεκμηριωμένη δυνατότητα τοπικής εγκατάστασης εξασφαλίζει τη δυνατότητα αξιολόγησης, ανάπτυξης και ενδεχόμενης ενσωμάτωσης της εφαρμογής από τρίτους.

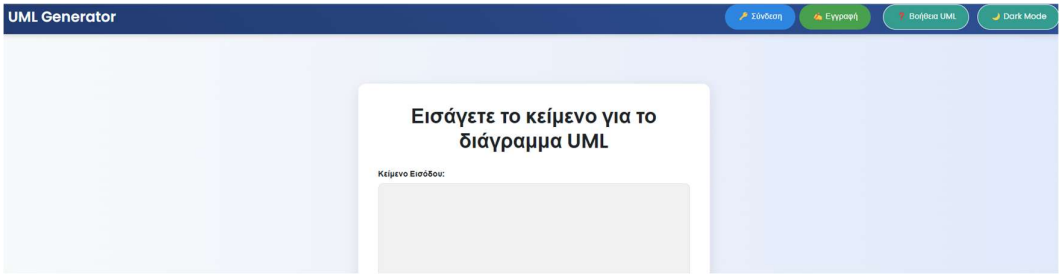
5.2 Επίδειξη Συστήματος

Στην παρούσα ενότητα παρουσιάζεται η λειτουργία της εφαρμογής μέσα από **ρεαλιστικά σενάρια χρήσης**, που αναδεικνύουν τις βασικές δυνατότητές της. Κάθε σενάριο συνοδεύεται από ενδεικτικά στιγμιότυπα (screenshots) της διεπαφής χρήστη (User Interface - UI) της εφαρμογής.

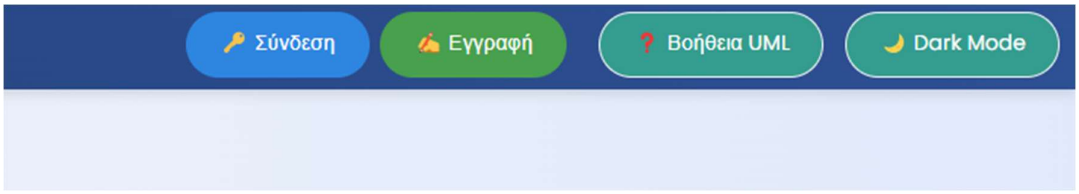
5.2.1 Σενάριο 1: Εγγραφή και Σύνδεση Χρήστη

Ο χρήστης μεταβαίνει στην **αρχική σελίδα της εφαρμογής**, όπως φαίνεται στην **Εικόνα 9** και επιλέγει το στοιχείο "**Εγγραφή**" από το μενού (δείτε **Εικόνα 10**). Στη συνέχεια, μεταφέρεται στη φόρμα δημιουργίας λογαριασμού, όπου **εισάγει** τα απαιτούμενα στοιχεία: όνομα χρήστη, email και κωδικό πρόσβασης (δείτε **Εικόνα 11**). Το σύστημα ελέγχει την εγκυρότητα των στοιχείων και, εφόσον είναι αποδεκτά, ολοκληρώνει την εγγραφή.

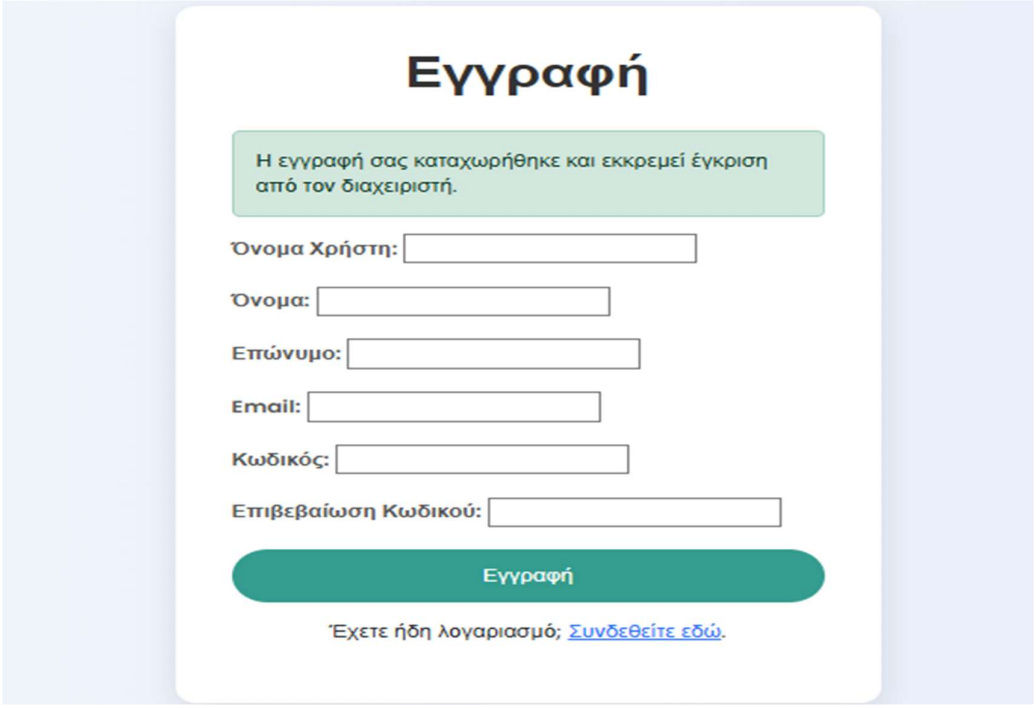
Μετά την επιτυχή εγγραφή, ο χρήστης πρέπει να περιμένει να του δώσει πρόσβαση ο διαχειριστής (admin). Έπειτα, θα μπορεί να μεταβεί στο στοιχείο **“Σύνδεση”**από το μενού της Εικόνας 10, και να συνδεθεί στην εφαρμογή χρησιμοποιώντας τα διαπιστευτήριά του (δείτε Εικόνα 12). Εφόσον συνδεθεί επιτυχώς, τότε του εμφανίζεται η αρχική σελίδα με επιπλέον δυνατότητες (δείτε Εικόνα 13).



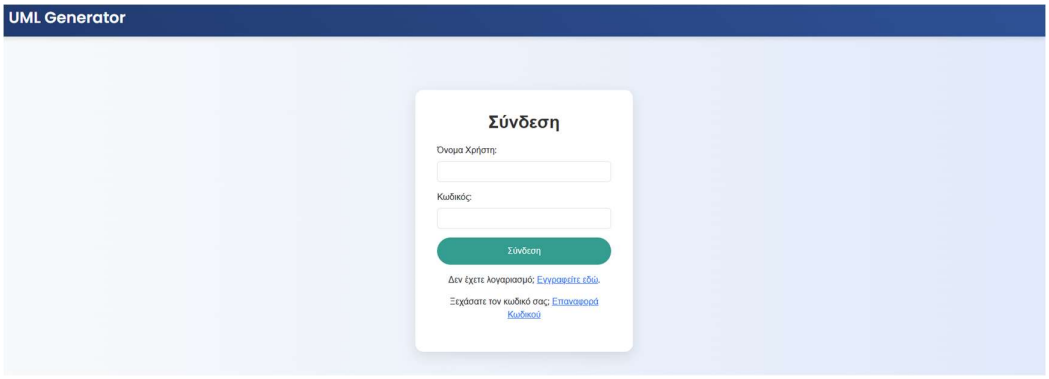
Εικόνα 9. Αρχική οθόνη εφαρμογής (μη συνδεδεμένου χρήστη)



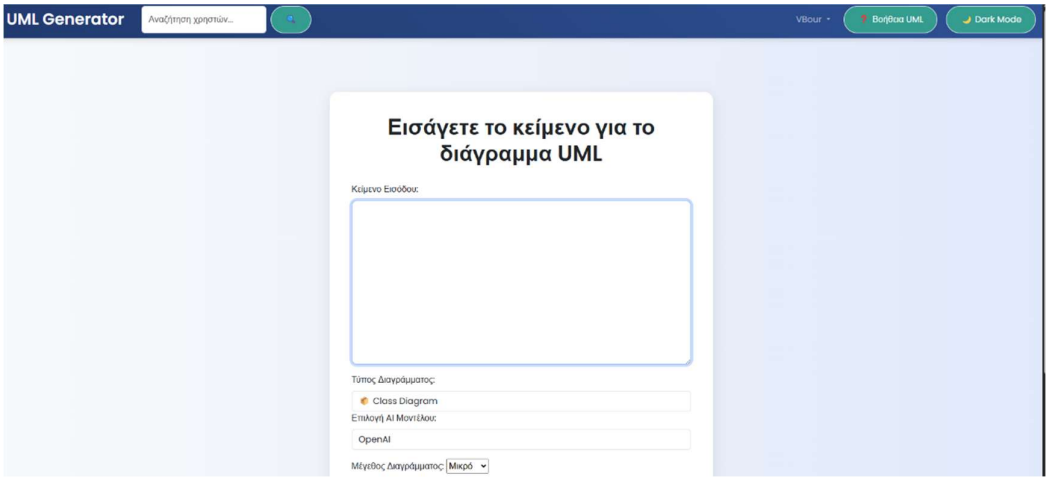
Εικόνα 10. Navigation bar (μη συνδεδεμένου χρήστη)



Εικόνα 11. Φόρμα εγγραφής



Εικόνα 12. Φόρμα σύνδεσης



Εικόνα 13. Αρχική οθόνη συνδεδεμένου χρήστη

5.2.2 Σενάριο 2: Δημιουργία Διαγράμματος από Περιγραφή

Το σενάριο χωρίζεται σε δύο εκδοχές, ανάλογα με το αν ο χρήστης είναι επισκέπτης ή συνδεδεμένος (εγγεγραμμένος) χρήστης.

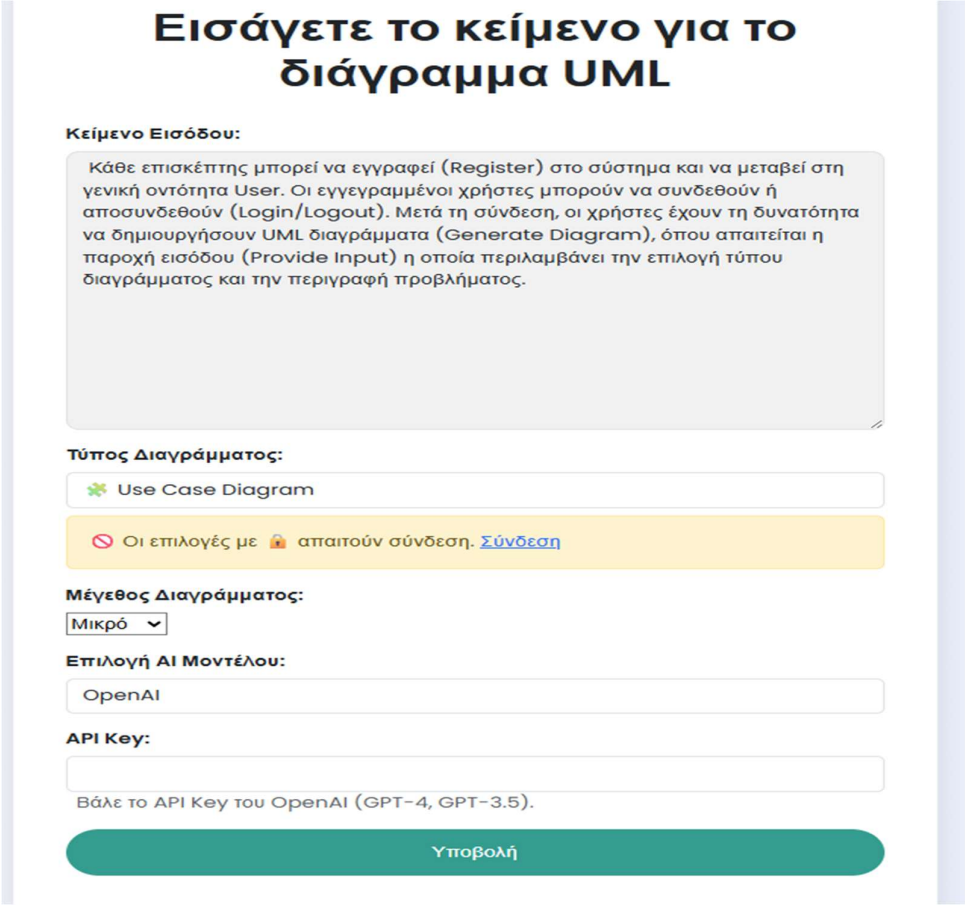
5.2.2.1 Επισκέπτης Χρήστης

Ο επισκέπτης (δηλαδή χρήστης που δεν έχει συνδεθεί ή δεν έχει λογαριασμό) έχει πρόσβαση σε **περιορισμένη εκδοχή της φόρμας εισαγωγής** (δείτε Εικόνα 14). Συγκεκριμένα:

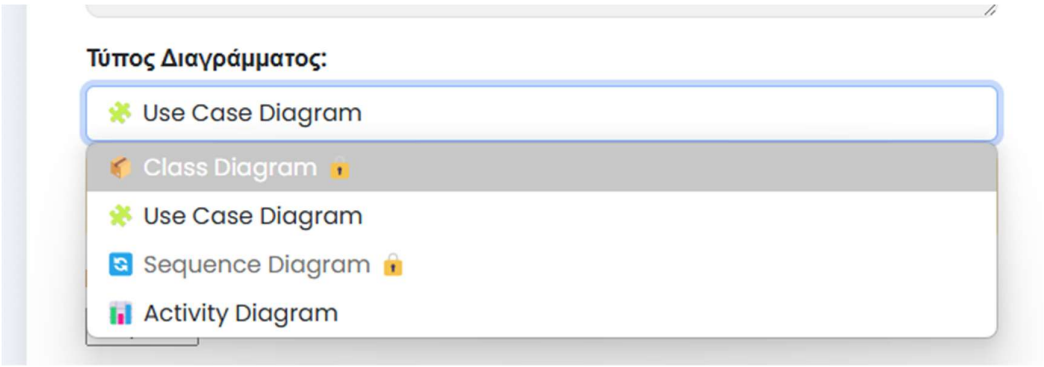
- Μπορεί να εισάγει **περιγραφή σε φυσική γλώσσα**
- Μπορεί να επιλέξει τύπο διαγράμματος (Use Case, Activity) (μόνο 2 από τα 4 που υποστηρίζονται συνολικά) (δείτε Εικόνα 14)
- Να κάνει λήψη του διαγράμματος σε PNG, JPEG και ASCII
- Μπορεί να κάνει επεξεργασία του διαγράμματος μέσα από το κώδικα UML

Αυτή η δυνατότητα απευθύνεται σε **γρήγορη δοκιμή λειτουργικότητας** από νέους χρήστες ή σε εκπαιδευτικά περιβάλλοντα.

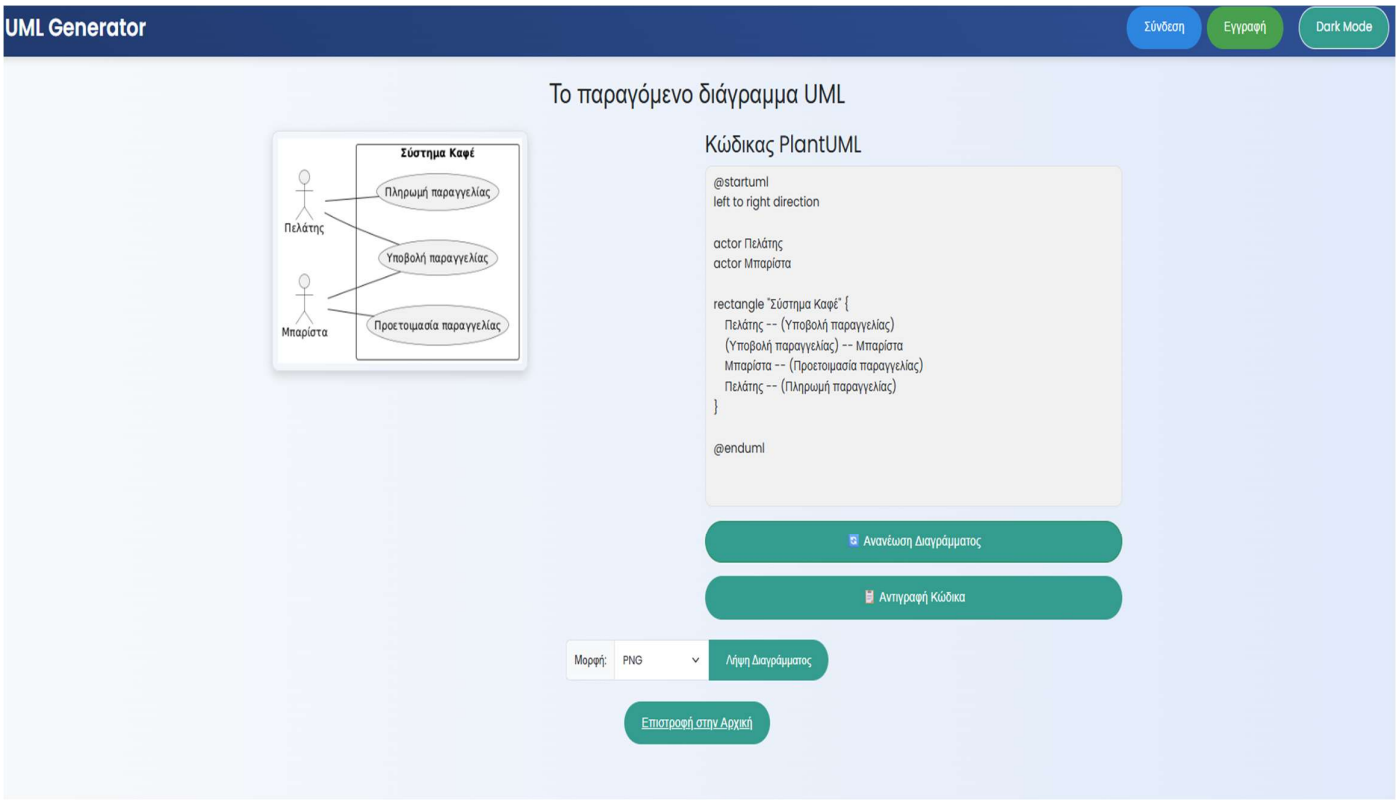
Η αρχική οθόνη (Εικόνα 14) επιτρέπει στον χρήστη να παρέχει την είσοδό του και να επιλέξει τύπο διαγράμματος (Εικόνα 15) καθώς και επιθυμητό μέγεθος. Επιπλέον, μπορεί να παρέχει το access token μόνο για το GPT-4 LLM (που προσφέρεται από τον πάροχο OpenAI). Εφόσον ο χρήστης πατήσει το κουμπί υποβολή, τότε του εμφανίζεται το διάγραμμα και ο PlantUML κώδικάς του (δείτε Εικόνα 16), τον οποίο μπορεί να αλλάξει . Εφόσον αλλάξει τον κώδικα, τότε πρέπει να πατήσει το κουμπί Ανανέωση Διαγράμματος για την αλλαγή της εμφάνισης του διαγράμματος.



Εικόνα 14. Αρχική οθόνη (μη συνδεδεμένου χρήστη)



Εικόνα 15. Επιλογές τύπου διαγράμματος (μη συνδεδεμένου χρήστη)



Εικόνα 16. Αποτέλεσμα υποβολής φόρμας

5.2.2.2 Εγγεγραμμένος Χρήστης

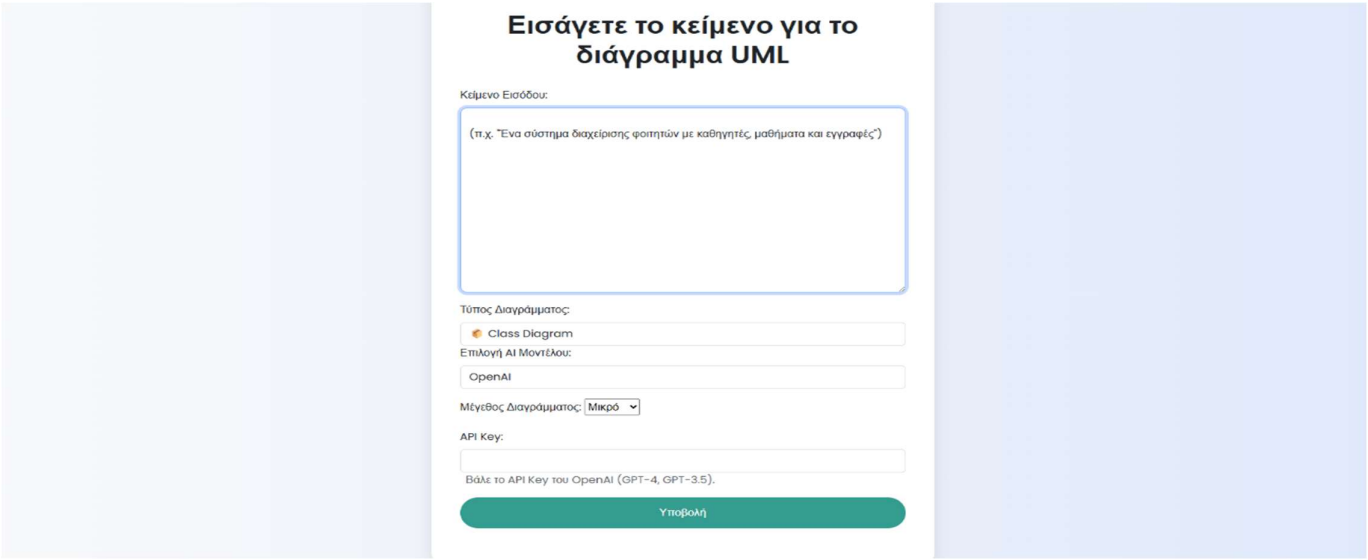
Μόλις ο χρήστης **συνδεθεί στο σύστημα**, αποκτά πρόσβαση στην **πλήρη φόρμα εισαγωγής** περιγραφής διαγράμματος (δείτε Εικόνα 17). Η φόρμα αυτή εμφανίζεται **αμέσως μετά τη σύνδεση** ή με την επιλογή «Νέο Διάγραμμα» από το μενού χρήστη.

Η φόρμα περιλαμβάνει τα εξής βήματα/στοιχεία:

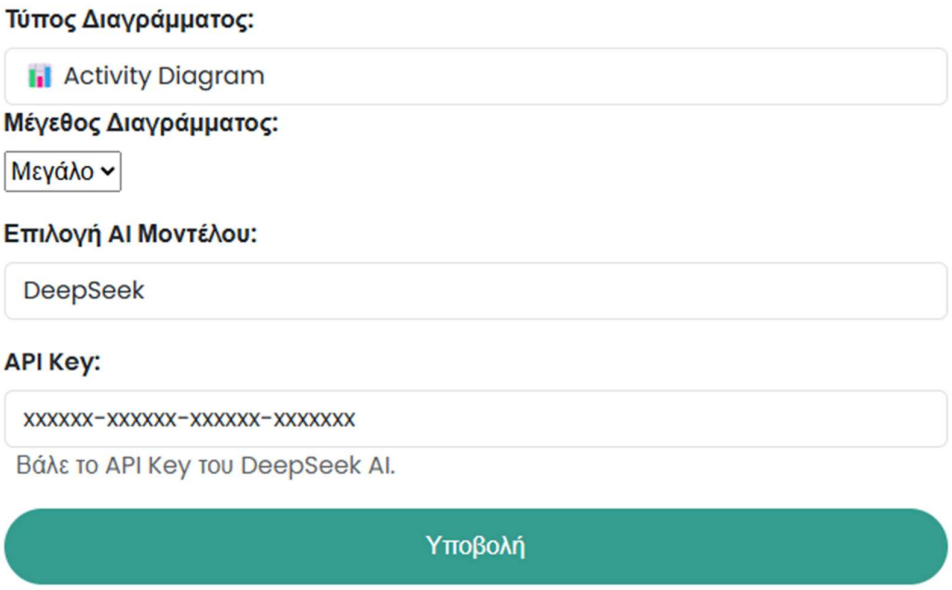
1. **Επιλογή τύπου διαγράμματος**: Class, Sequence, Use Case ή Activity
2. **Επιλογή μεγέθους εξόδου**: Μικρό ή Μεγάλο
3. **Εισαγωγή περιγραφής** σε φυσική γλώσσα
4. **Επιλογή LLM** (GPT, Gemini, DeepSeek)
5. **Παροχή API key** για χρήση του αντίστοιχου LLM (Το API key παρέχεται από τον χρήστη κατά τη δημιουργία του κάθε UML αιτήματος, μέσω της φόρμας UMLInputForm. Δεν αποθηκεύεται μόνιμα στο προφίλ ή σε σχετική κλάση LLM.
   Αυτό ενισχύει την ασφάλεια, καθώς αποφεύγεται η αποθήκευση ευαίσθητων tokens στη βάση δεδομένων.).

Στη συνέχεια, εφόσον ο χρήστης παρέχει την κατάλληλη είσοδο (δείτε Εικόνα 18), πατάει "Υποβολή" και το σύστημα:

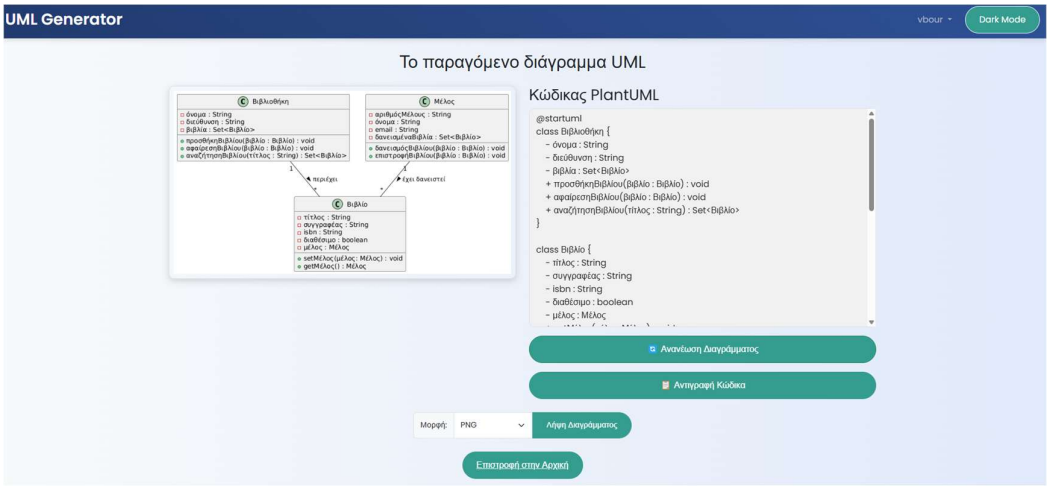
- δημιουργεί το κατάλληλο prompt με βάση την παρεχόμενη είσοδο του χρήστη, καλεί το LLM, και επιστρέφει το αντίστοιχο διάγραμμα (Εικόνα 19).
- Ο συνδεδεμένος χρήστης μπορεί επιπλέον (όπως φαίνεται από την Εικόνα 19):
  - Να επεξεργαστεί το μοντέλο PlantUML που έχει παραχθεί, όπως αναφέρθηκε και για τον επισκέπτη
  - Να εξάγει το διάγραμμα σε PNG, SVG ή ASCII



Εικόνα 17. Αρχική οθόνη συνδεδεμένου χρήστη με κείμενο εισόδου



Εικόνα 18. Επιλογές συνδεδεμένου χρήστη για την δημιουργία διαγράμματος



Εικόνα 19. Οθόνη τελικού αποτλέσματος

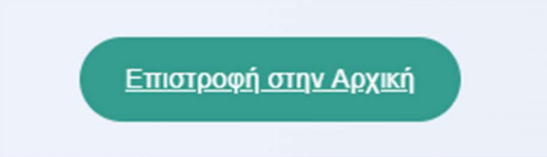
5.2.3 Σενάριο 3: Τροποποίηση Παραγόμενου Διαγράμματος

Εφόσον ο χρήστης επιθυμεί να τροποποιήσει το παραγόμενο διάγραμμα, η εφαρμογή του προσφέρει δύο εναλλακτικές διαδρομές:

5.2.3.1 Νέα Υποβολή Περιγραφής

- Ο χρήστης επιστρέφει στην αρχική σελίδα ή στη φόρμα παραγωγής διαγράμματος πατώντας το κουμπί "Επιστροφή στην Αρχική" (δείτε κάτω

μέρος Εικόνας 19 και Εικόνα 20) ή το logo που βρίσκεται αριστερά πάνω στο navigation bar "UML Generator" και λειτουργεί και ως home button (δείτε πάνω αριστερό μέρος της Εικόνας 19 και την Εικόνα 21).
- Εισάγει νέα περιγραφή.
- Επιλέγει ξανά τις παραμέτρους παραγωγής (τύπος διαγράμματος, LLM, μέγεθος) και υποβάλλει το νέο αίτημα.



Εικόνα 20. Κουμπί Επιστροφής στην αρχική σελίδα

Εικόνα 21. Logo που επιτρέπει την επιστροφή στην αρχική σελίδα εφόσον πατηθεί

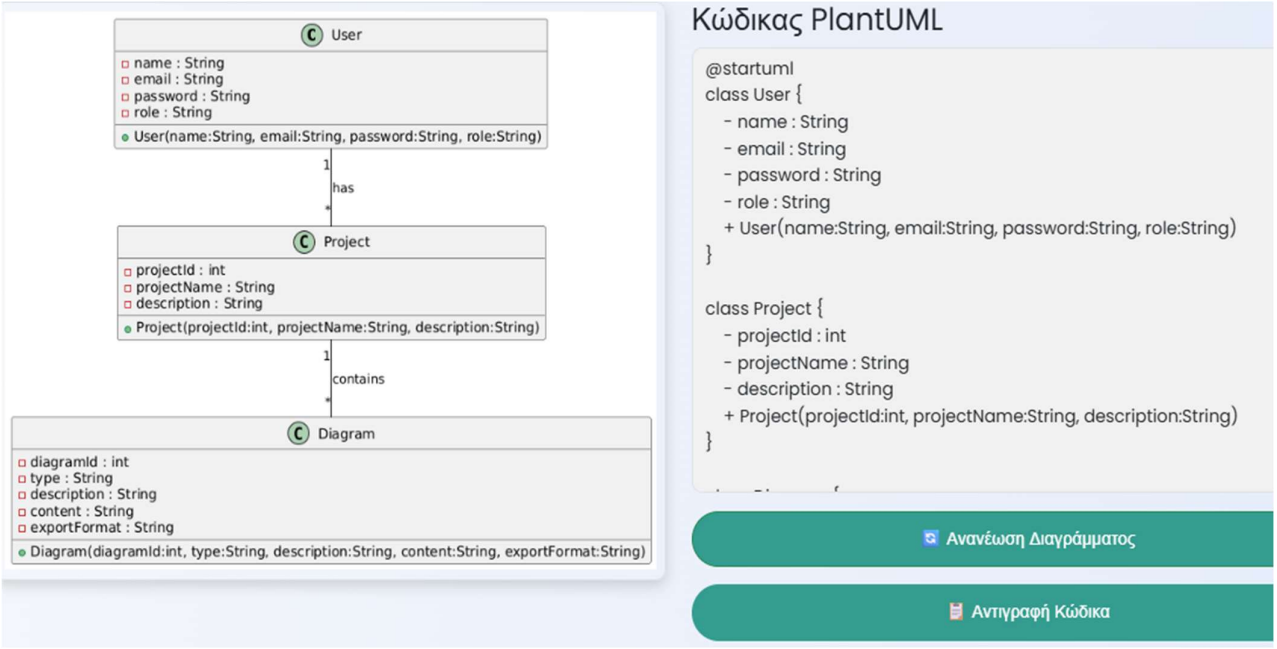
Αυτή η ροή είναι χρήσιμη όταν ο χρήστης δεν είναι ικανοποιημένος με το αποτέλεσμα (δηλ. το τρέχων περιεχόμενο διαγράμματος) ή πρέπει να γίνουν σημαντικές αλλαγές στο διάγραμμα αυτό.

5.2.3.2 Άμεση Επεξεργασία του UML Κώδικα

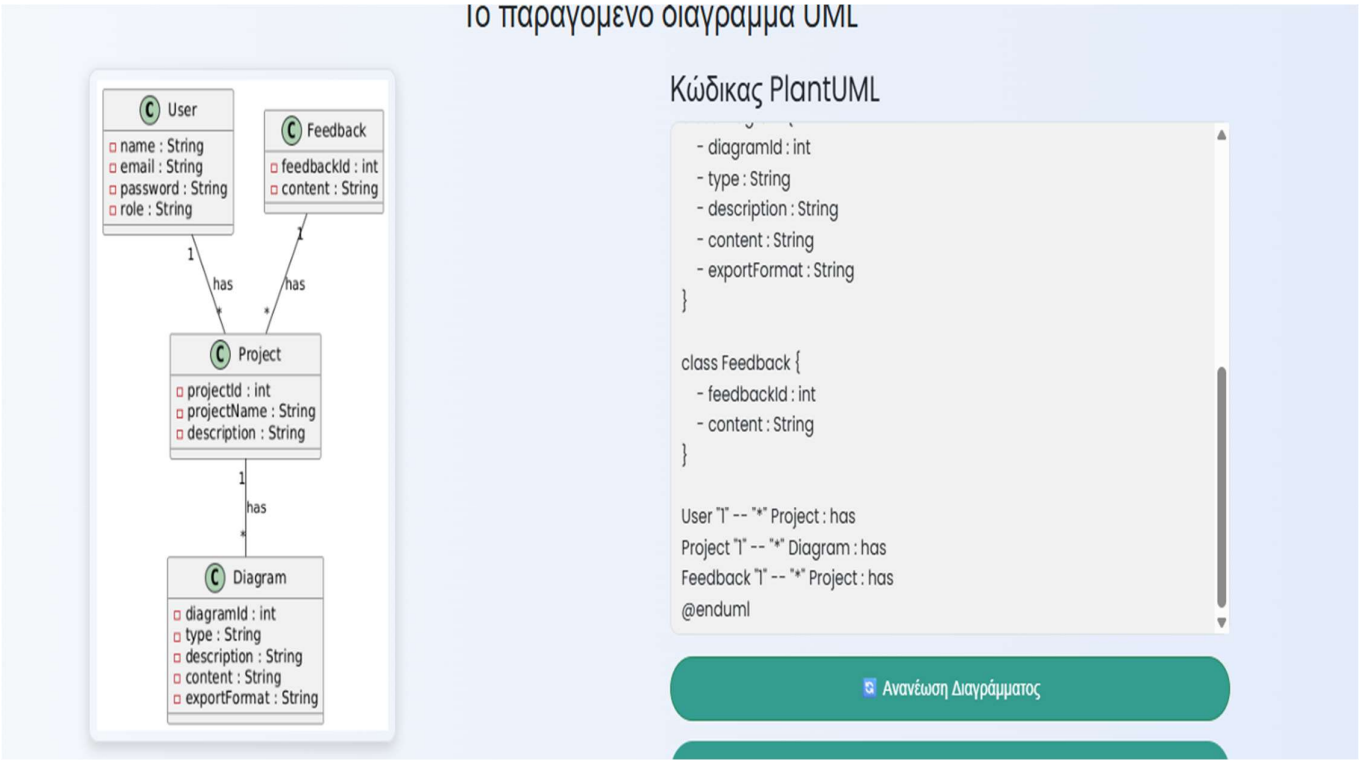
- Ο χρήστης τροποποιεί απευθείας τον παραγόμενο PlantUML κώδικα μέσω του ενσωματωμένου editor (δείτε Εικόνα 22)
- Με την επιλογή **"Ανανέωση Κώδικα"**, η εφαρμογή ενημερώνει το σχήμα με βάση τον νέο UML κώδικα (δείτε Εικόνα 23), **χωρίς επαναποστολή σε LLM**.

Αυτή η ροή είναι χρήσιμη όταν ο χρήστης εντοπίζει μικρές αλλαγές που μπορούν να γίνουν άμεσα και τοπικά (π.χ. διόρθωση ονόματος, προσθήκη συσχέτισης), χωρίς την ανάγκη επανα-χρησιμοποίησης LLM.

Και στις δύο περιπτώσεις (νέα υποβολή περιγραφής & άμεση επεξεργασία UML κώδικα), το σύστημα **ενημερώνει τη διεπαφή χρήστη σε πραγματικό χρόνο**, ώστε να αποτυπωθούν οι (απαιτούμενες) αλλαγές άμεσα στο διάγραμμα.



Εικόνα 22. Χρήση PlantUML editor για την τροποποίηση του διαγράμματος



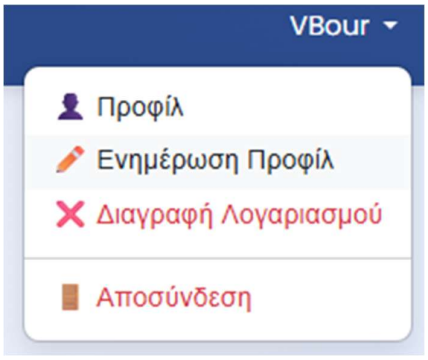
Εικόνα 23. Παραγόμενο διάγραμμα μετά την τροποποίηση του κώδικά του

5.2.4 Σενάριο 4: Επεξεργασία Προφίλ Χρήστη

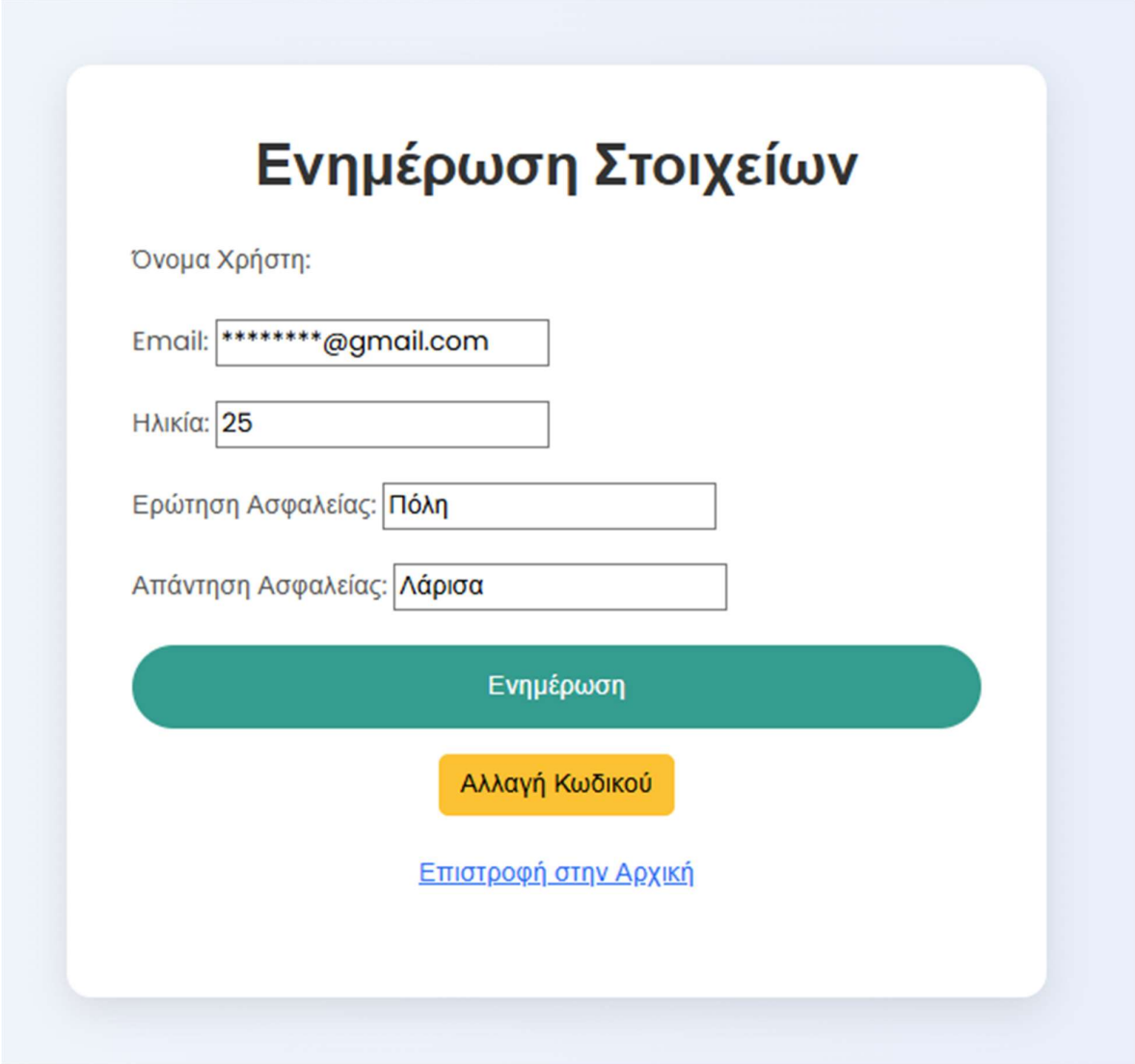
Ο χρήστης μπορεί να επεξεργαστεί τα στοιχεία του προφίλ του επιλέγοντας το drop down list όπου ανοίγει κάνοντας κλικ το όνομα χρήστη του (δείτε Εικόνα 24) στο πάνω δεξιά μέρος της οθόνης/σελίδας. Κατά την είσοδό του στη φόρμα επεξεργασίας (δείτε Εικόνα 25), προβάλλονται **οι ήδη αποθηκευμένες τιμές** των πεδίων (όπως ηλικία, ερώτηση ασφαλείας), ώστε ο χρήστης να μπορεί να τις επιθεωρήσει και να αποφασίσει αν χρειάζεται να τις τροποποιήσει.

Συγκεκριμένα, ο χρήστης έχει τις εξής δυνατότητες:

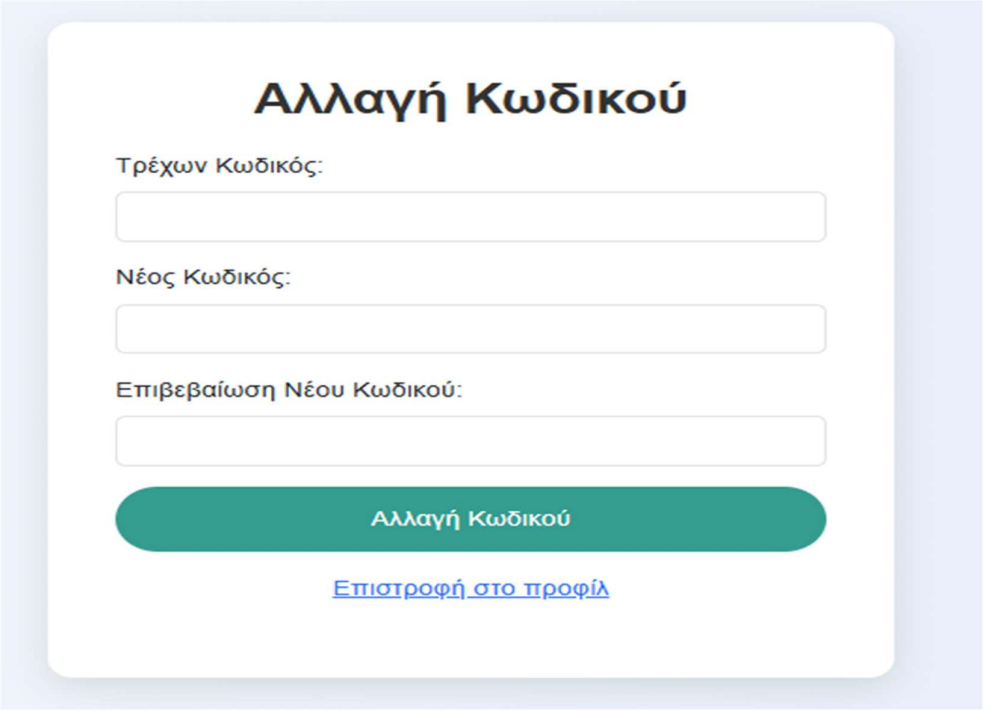
- **Ενημέρωση ηλικίας**: Το πεδίο είναι αριθμητικό και εμφανίζει την τρέχουσα τιμή. Ο χρήστης μπορεί να εισάγει νέα τιμή και να την αποθηκεύσει.
- **Ρύθμιση ερώτησης ασφαλείας:** Ο χρήστης μπορεί να πληκτρολογήσει δική του ερώτηση, αλλά απαιτείται προσοχή ώστε η ερώτηση να είναι ασφαλής και προσωπική.
- **Αλλαγή απάντησης ασφαλείας**: Το πεδίο εμφανίζεται κενό για λόγους ασφαλείας. Ο χρήστης μπορεί να το ενημερώσει μαζί με την ερώτηση ή ανεξάρτητα.
- **Αλλαγή κωδικού πρόσβασης**: Η φόρμα παρέχει ξεχωριστή ενότητα για αλλαγή κωδικού. Εφόσον ο χρήστης πατήσει το κουμπί “Αλλαγή Κωδικού”, του εμφανίζεται ξεχωριστή φόρμα (δείτε Εικόνα 26) όπου θα πρέπει πρώτα να εισάγει τον παλαιό του κωδικό και έπειτα να εισάγει τον νέο κωδικό δύο φορές για επιβεβαίωση.



Εικόνα 24. Dropdown list του Navigation bar



Εικόνα 25. Φόρμα Ενημέρωσης στοιχείων

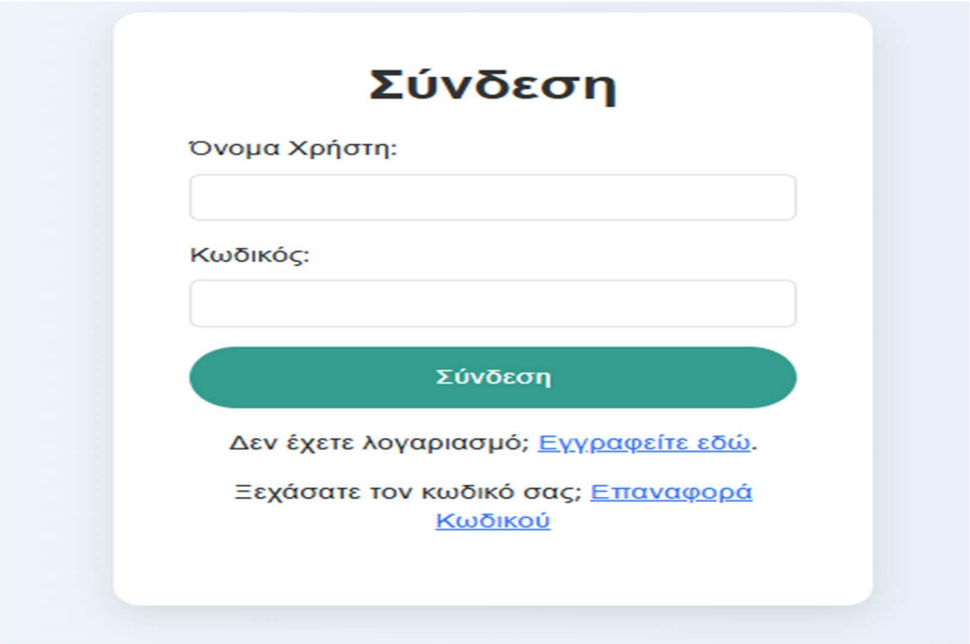


Εικόνα 26. Φόρμα αλλαγής κωδικού

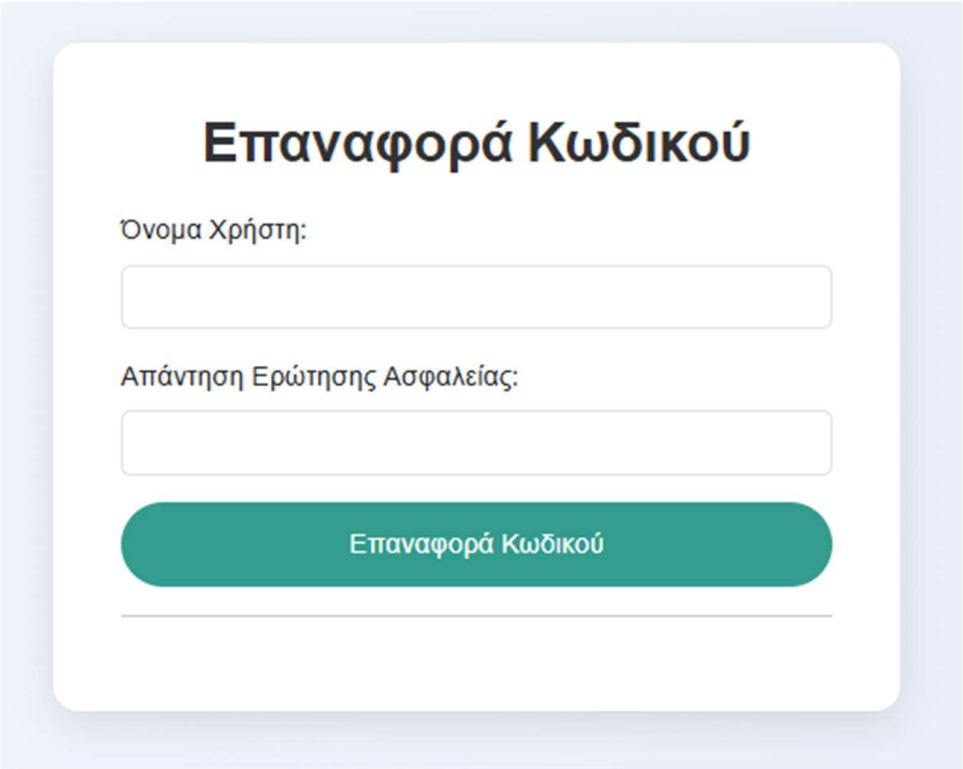
5.2.5 Σενάριο 5: Επαναφορά κωδικού

Εφόσον ο χρήστης ξεχάσει τον κωδικό του μπορεί να τον επαναφέρει από την φόρμα σύνδεσης όπου του δίνεται και η επιλογή "Επαναφορά κωδικού" (δείτε Εικόνα 27). Με αυτήν την επιλογή θα του ανοίξει μία φόρμα (δείτε Εικόνα 28) όπου ο χρήστης θα πρέπει να βάλει το όνομα χρήστη και την απάντηση από την ερώτηση ασφαλείας. Στην συνέχεια, σε περίπτωση επιτυχούς συμπλήρωσης και υποβολής των 2 προαναφερόμενων στοιχείων (δείτε Εικόνα 29), θα του έρθει ένα email στον λογαριασμό email που έχει δηλώσει με την εγγραφή του, το οποίο θα περιλαμβάνει τον προσωρινό κωδικό του που θα διατηρηθεί για 3 ημέρες (δείτε Εικόνα 30). Για να συνδεθεί θα χρησιμοποιήσει το όνομα χρήστη και τον προσωρινό κωδικό του ως password. Εάν υπάρχει **ενεργή αίτηση επαναφοράς κωδικού**, εμφανίζεται προειδοποιητικό μήνυμα (δείτε Εικόνα 31) που ενημερώνει τον χρήστη για τον χρόνο που απομένει (για να αλλάξει τον κωδικό του) πριν την αυτόματη απενεργοποίηση του λογαριασμού, σύμφωνα με την πολιτική ασφάλειας της εφαρμογής.

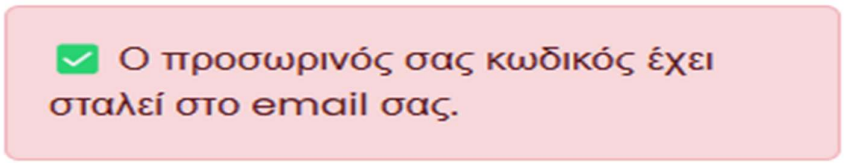
Η διεπαφή είναι απλή, καθοδηγητική και ενημερώνει τον χρήστη σε πραγματικό χρόνο για το αν η κάθε ενέργεια ολοκληρώθηκε επιτυχώς ή όχι.



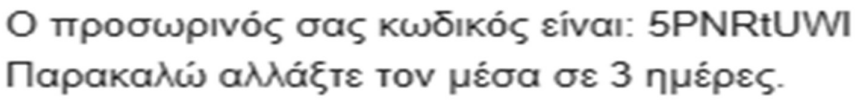
Εικόνα 27. Φόρμα σύνδεσης



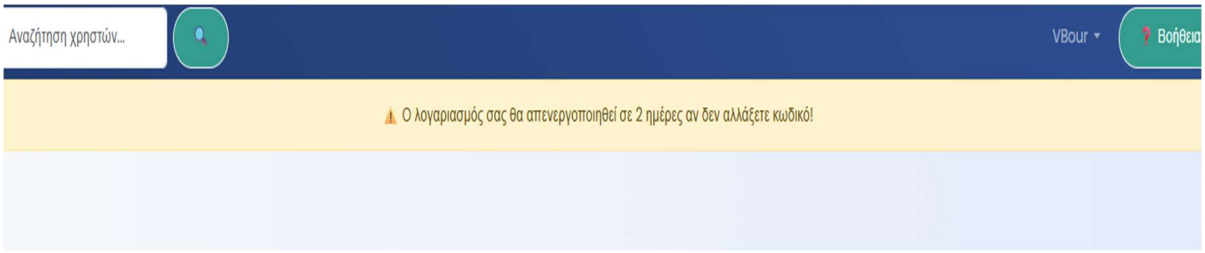
Εικόνα 28. Φόρμα επαναφοράς κωδικού



Εικόνα 29. Μήνυμα ότι ο προσωρινός κωδικός έχει σταλεί στο email



Εικόνα 30. Μήνυμα που έρχεται στο email χρήστη



Εικόνα 31. Υπενθύμιση απενεργοποίησης λογαριασμού μετά από επαναφορά κωδικού

5.2.6 Σενάριο 6: Αναζήτηση Χρηστών από Διαχειριστή

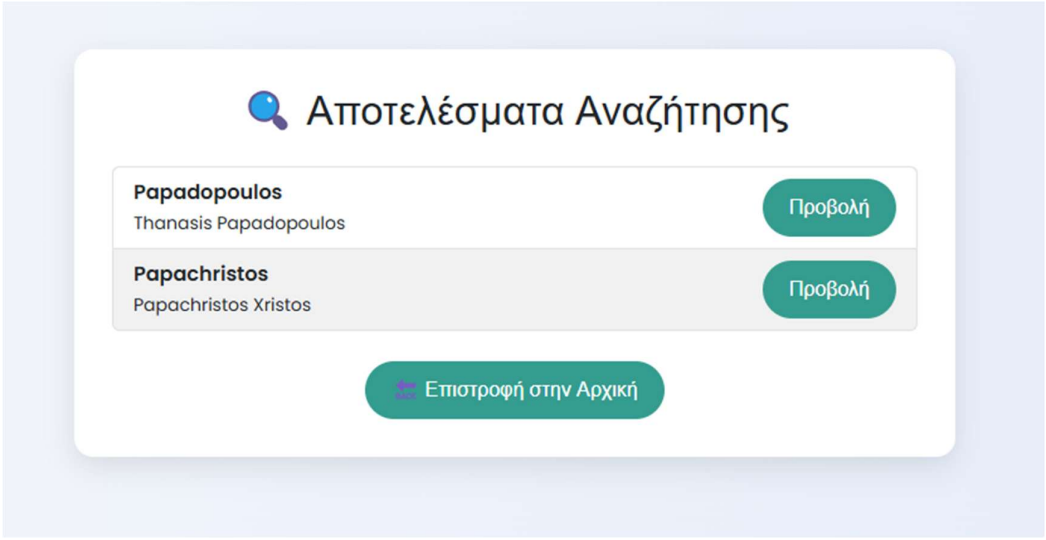
Ο διαχειριστής της εφαρμογής διαθέτει πρόσβαση σε ειδική διεπαφή αναζήτησης χρηστών μέσω custom admin dashboard, το οποίο έχει υλοποιηθεί εντός της εφαρμογής και είναι προσβάσιμο αποκλειστικά από χρήστες με ρόλο admin. Η πρόσβαση για την προβολή των χρηστών από τον διαχειριστή γίνεται μέσω της μπάρας αναζήτησης που έχει ο admin στο navigation bar.

Μέσω αυτής της διεπαφής, ο διαχειριστής μπορεί να πραγματοποιεί αναζήτηση χρηστών (δείτε Εικόνα 32) με βάση φίλτρα, όπως username, και να προβάλλει βασικές πληροφορίες για κάθε αποτέλεσμα. Στην Εικόνα 33, παρουσιάζονται τα αποτελέσματα αναζήτησης για το keyword "Pap", με βάση το οποίο εντοπίστηκαν δύο χρήστες.

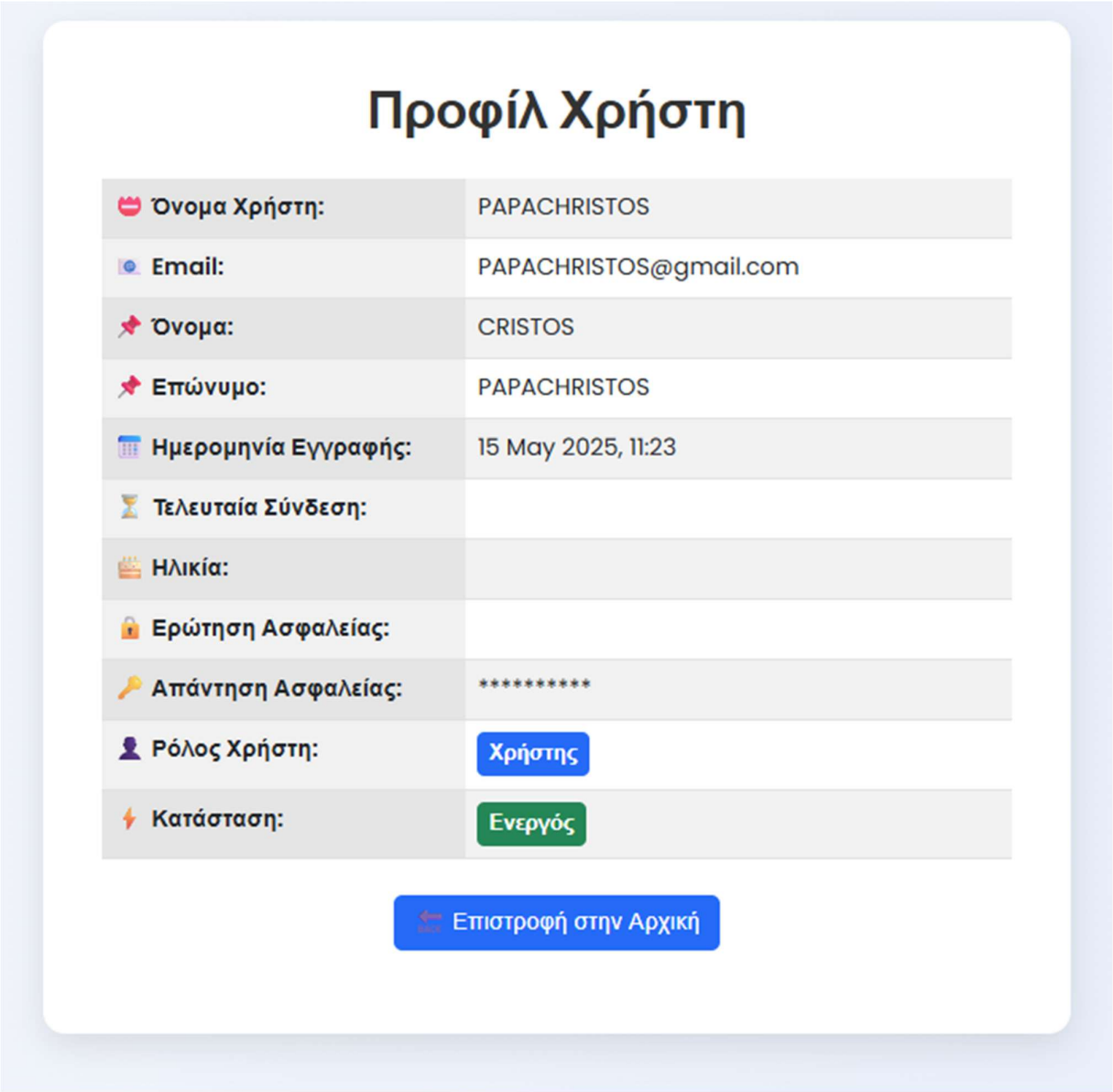
Για κάθε χρήστη εμφανίζεται κουμπί «Προβολή», το οποίο οδηγεί σε ξεχωριστή σελίδα (δείτε Εικόνα 34) με αναλυτικά στοιχεία του χρήστη (π.χ. όνομα, email, username, ημερομηνία εγγραφής κ.λπ.). Ωστόσο, στο custom dashboard δεν υποστηρίζεται τροποποίηση ή διαγραφή των χρηστών.

Για την εκτέλεση τέτοιων ενεργειών (π.χ. αλλαγή στοιχείων, διαγραφή λογαριασμού, αλλαγή ρόλου), ο διαχειριστής μεταβαίνει στο Django Admin Panel (http://127.0.0.1:8000/admin/) , το οποίο προσφέρει πλήρεις δυνατότητες επεξεργασίας χρηστών, σύμφωνα με τις ενσωματωμένες λειτουργίες του Django (δείτε Εικόνες 35 & 36).

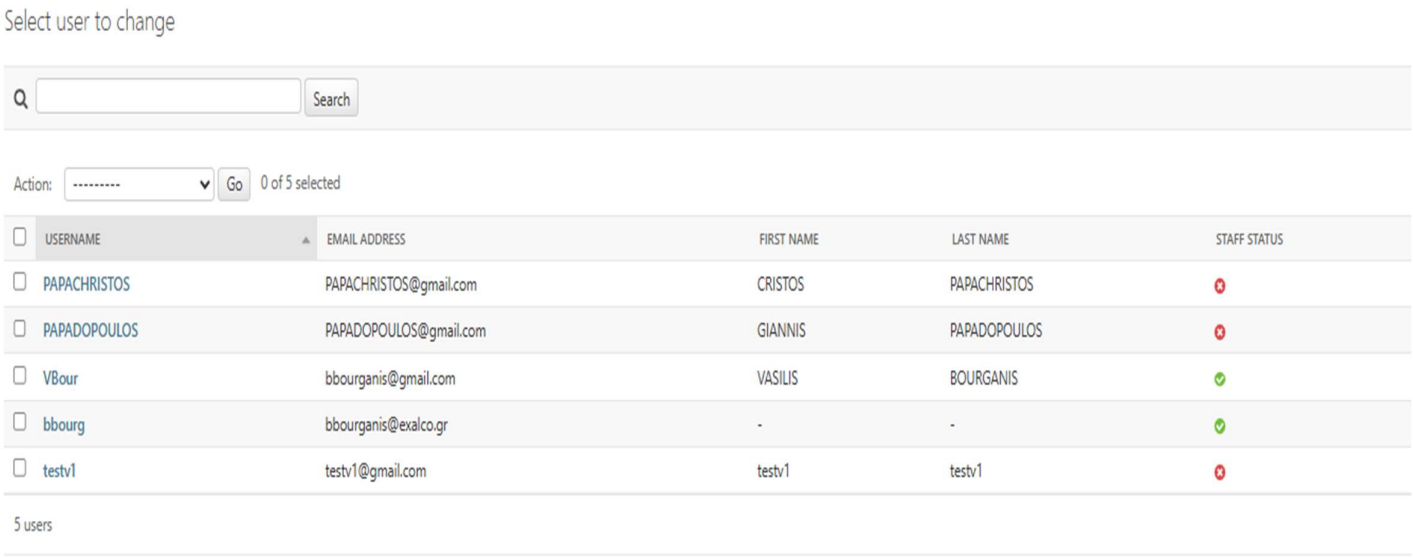
Εικόνα 32. Admin search bar



Εικόνα 33. Αποτελέσματα της αναζήτησης της Εικόνας 32



Εικόνα 34. Προβολή προφίλ του χρήστη που επιλέχθηκε μέσω του custom search bar



Εικόνα 35. Django Admin Panel

Εικόνα 36. Προβολή χρήστη στο Admin panel του Django

Κεφάλαιο 6 – Αποτίμηση Συστήματος

6.1 Σχεδίαση & Εκτέλεση Αποτίμησης

Η αποτίμηση του συστήματος πραγματοποιήθηκε με σκοπό να αξιολογηθεί η **ακρίβεια και η απόδοση** της εφαρμογής, καθώς και η **απόδοση των διαφορετικών LLMs** στη διαδικασία παραγωγής UML διαγραμμάτων. Οι στόχοι της αποτίμησης περιλαμβάνουν τον έλεγχο της ποιότητας των παραγόμενων διαγραμμάτων σε σχέση με τις εισόδους και τον χρόνο απόκρισης του συστήματος.

Η αποτίμηση στηρίχθηκε σε **πειραματική μεθοδολογία**, με έλεγχο της λειτουργικότητας και απόδοσης του συστήματος σε πραγματικά σενάρια χρήσης. Χρησιμοποιήθηκαν τέσσερις τυπικές περιγραφές προβλημάτων, σε μορφή φυσικής γλώσσας, οι οποίες καλύπτουν κοινές περιπτώσεις μοντελοποίησης, όπως διαχείριση χρηστών. Κάθε περιγραφή αντιστοιχήθηκε σε έναν συγκεκριμένο τύπο διαγράμματος (Class, Sequence, Use Case ή Activity) και μοντελοποιήθηκε με τη χρήση ενός LLM (ChatGPT, Gemini ή DeepSeek). Με τον τρόπο αυτό, καλύφθηκαν όλα τα είδη διαγραμμάτων που υποστηρίζει η εφαρμογή.

Το σύνολο δεδομένων δημιουργήθηκε αποκλειστικά για τους σκοπούς της παρούσας αποτίμησης και περιλαμβάνει, εκτός από τις περιγραφές προβλημάτων, και το αντίστοιχο ιδανικό διάγραμμα επίλυσης (gold standard). Τα διαγράμματα αυτά κατασκευάστηκαν από τον συγγραφέα της εργασίας με βάση τις βέλτιστες πρακτικές μοντελοποίησης UML, ώστε να λειτουργήσουν ως σημείο αναφοράς για την αξιολόγηση της ποιότητας των παραγόμενων διαγραμμάτων. Κατά την εκτέλεση της αποτίμησης, συλλέχθηκαν δεδομένα σχετικά με:

- Την **ορθότητα και πληρότητα** του παραγόμενου διαγράμματος, σε σχέση με την αρχική περιγραφή χρήστη - αυτό καλύπτει τη σημασιολογική εγκυρότητα της εξόδου / διαγράμματος
- Την **συντακτική εγκυρότητα** της εξόδου σε μορφή PlantUML
- Τον **χρόνο εκτέλεσης** όσον αφορά την λειτουργία της παραγωγής διαγράμματος της εφαρμογής, ο οποίος μετρήθηκε από τη στιγμή λήψης του αιτήματος του χρήστη από το backend έως την παραγωγή του διαγράμματος από το backend. Για τον υπολογισμό του χρόνου εκτέλεσης, υλοποιήθηκε αντίστοιχος κώδικας που αναλύεται παρακάτω. Ακολουθεί ένα παράδειγμα εκτέλεσής του.

```
Χρόνος παραγωγής διαγράμματος: 1.47 δευτερόλεπτα
[14/May/2025 21:08:18] "POST / HTTP/1.1" 302 0
[14/May/2025 21:08:18] "GET /result/ HTTP/1.1" 200 8972
```

Εικόνα 37. Αποτελέσματα του terminal σχετικά με την εκτύπωση του χρόνου παραγωγής του διαγράμματος

Μετρικές Αξιολόγησης

Για την αποτίμηση της απόδοσης του συστήματος χρησιμοποιήθηκαν συγκεκριμένες **μετρικές**, οι οποίες αποδίδουν ποσοτικά την ποιότητα της εξόδου και αξιολογούν τη συμπεριφορά του συστήματος σε πραγματικά σενάρια χρήσης. Οι μετρικές αυτές προσδιορίστηκαν ώστε να καλύπτουν τόσο την **τεχνική εγκυρότητα**, όσο και τη **σημασιολογική ορθότητα** της παραγόμενης πληροφορίας καθώς και την χρονική απόδοση του συστήματος.

1. Σημασιολογική Ορθότητα (Semantic Correctness)

Η **σημασιολογική ορθότητα** εκτιμά κατά πόσο το παραγόμενο UML διάγραμμα αναπαριστά **εννοιολογικά σωστά** την περιγραφή του χρήστη, σε σύγκριση με ένα πρότυπο διάγραμμα αναφοράς (gold standard). Για τον υπολογισμό της ιδιότητας αυτής χρησιμοποιούνται τρεις μετρικές:

1.1 Ακρίβεια (Precision)

Αξιολογεί το ποσοστό των σωστά εντοπισμένων στοιχείων σε σχέση με όλα τα παραγόμενα από το σύστημα.

Τύπος:

Precision = TP / (TP + FP)

- **TP (True Positives):** Τα σωστά παραγόμενα στοιχεία που υπάρχουν και στο πρότυπο διάγραμμα.
- **FP (False Positives):** Τα επιπλέον ή λανθασμένα στοιχεία που δεν υπήρχαν στο πρότυπο.

Υψηλή τιμή Precision σημαίνει ελάχιστος "θόρυβος" στην έξοδο (δηλ. χωρίς πολλά περιττά ή ανακριβή στοιχεία).

1.2 Ανάκληση (Recall – Πληρότητα)

Εκφράζει το ποσοστό των στοιχείων του gold standard που **όντως εντοπίστηκαν** από το σύστημα.

Τύπος:

Recall = TP / (TP + FN)

- **FN (False Negatives):** Στοιχεία που υπήρχαν στο πρότυπο διάγραμμα αλλά παραλείφθηκαν από το σύστημα.

Υψηλό Recall συνεπάγεται σχεδόν πλήρη κάλυψη της απαιτούμενης πληροφορίας από το σύστημα.

1.3 F1-Score (Αρμονικός μέσος)

Συνδυάζει τις δύο προηγούμενες μετρικές σε έναν ενιαίο δείκτη.

Τύπος:

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall)$$

Το F1-score παρέχει ισορροπημένη εκτίμηση της σημασιολογικής ορθότητας, λαμβάνοντας υπόψη τόσο την ακρίβεια όσο και την πληρότητα της εξόδου.

2. Συντακτική Ορθότητα (Syntactic Validity)

Η μετρική αυτή αξιολογεί κατά πόσο το παραγόμενο **PlantUML script** είναι **συντακτικά έγκυρο** – δηλαδή αν μπορεί να αποδοθεί από τον PlantUML renderer χωρίς σφάλματα σύνταξης.

Τύπος:

$$Syntactic\ Validity = N_{valid} / N_{total}$$

- **N_valid**: Αριθμός διαγραμμάτων που είναι συντακτικά έγκυρα.
- **N_total**: Συνολικός αριθμός περιπτώσεων αξιολόγησης (prompts).

Σημείωση: Λαμβάνεται υπόψη μόνο το τελικό αποτέλεσμα (ανεξάρτητα από τον αριθμό αποτυχημένων προσπαθειών). Ενδιάμεσες αποτυχημένες απόπειρες δεν επηρεάζουν τη βαθμολογία.

3. Χρόνος Εκτέλεσης (Execution Time)

Η μετρική αυτή μετρά τη **χρονική καθυστέρηση** από τη στιγμή που το backend λαμβάνει το αίτημα του χρήστη έως τη στιγμή που παράγεται το τελικό αποτέλεσμα (διάγραμμα σε εικόνα και κώδικα).

Τύπος μέτρησης:

```
end_time = time.time()

execution_time = end_time - start_time
```

Ο υπολογισμός βασίζεται σε **χρονικές σφραγίδες (timestamps)** εντός του backend. Ο μέσος χρόνος εκτέλεσης υπολογίστηκε από όλες τις περιπτώσεις (παραγωγής διαγράμματος) που επιτελέστηκαν στο πλαίσιο της αξιολόγησης του συστήματος και καταγράφηκε σε δευτερόλεπτα.

- Η μετρική αυτή αξιολογεί τη **χρονική απόδοση του συστήματος** και τη δυνατότητα **real-time αλληλεπίδρασης**.

6.2 Αποτελέσματα Αποτίμησης

Στο παρόν υποκεφάλαιο παρατίθεται η αποτίμηση της λειτουργικότητας της εφαρμογής μέσω της αξιολόγησης των παραγόμενων διαγραμμάτων UML. Για κάθε κατηγορία διαγράμματος που υποστηρίζεται από το σύστημα (Class, Use Case, Activity, Sequence), αξιοποιήθηκε ένα σενάριο σε φυσική γλώσσα, το οποίο συντάχθηκε ειδικά για τις ανάγκες της παρούσας αποτίμησης. Με βάση αυτό το prompt, παραγόταν το αντίστοιχο διάγραμμα μέσω LLM και συγκρινόταν με ένα αντίστοιχο **πρότυπο (gold standard)** διάγραμμα, το οποίο δημιουργήθηκε **από τον συγγραφέα της παρούσας εργασίας** αξιοποιώντας τεκμηριωμένες γνώσεις UML και βέλτιστες πρακτικές μοντελοποίησης.

Η διαδικασία αυτή επέτρεψε τη συστηματική αξιολόγηση των εξόδων του συστήματος, με βάση τη κοινή περιγραφή προβλήματος, ώστε να υπολογιστούν μετρικές, όπως η σημασιολογική **ακρίβεια** και **πληρότητα/ανάκληση**. Αν και τα gold standard διαγράμματα δεν θεωρούνται απόλυτα "σωστά", καθώς το ίδιο σενάριο μπορεί να μοντελοποιηθεί με περισσότερους από έναν σωστούς τρόπους, λειτούργησαν ως σταθερό σημείο αναφοράς για συγκρίσεις με αντικειμενικό και επαναλήψιμο τρόπο.

Κατά τη δημιουργία του διαγράμματος, ο χρήστης επέλεγε τρεις βασικές παραμέτρους: τον τύπο του διαγράμματος (Class, Sequence, Use Case ή Activity), το LLM μοντέλο (GPT-3.5, Gemini, DeepSeek), καθώς και το επιθυμητό μέγεθος εξόδου (Μικρό ή Μεγάλο). Οι επιλογές αυτές εισάγονταν μέσω της διεπαφής και ενσωματώνονταν δυναμικά στο prompt προς το LLM.

Για τις περισσότερες δοκιμές επιλέχθηκε το μοντέλο **Gemini**, κυρίως λόγω της **σταθερής λειτουργίας του API**, της ευκολίας ενσωμάτωσης και της άμεσης προσβασιμότητας χωρίς ανάγκη επιπλέον ρυθμίσεων. Παρόλο που και άλλα LLMs όπως το **DeepSeek** είναι προσβάσιμα δωρεάν μέσω τρίτων πλατφορμών (όπως HuggingFace ή OpenRouter), η χρήση τους απαιτεί επιπλέον παραμετροποίηση ή εναλλακτικές βιβλιοθήκες ενσωμάτωσης, που δεν καλύπτονταν πλήρως στο τρέχον στάδιο υλοποίησης της εφαρμογής.

Στη συνέχεια, παρουσιάζουμε τα αποτελέσματα αποτίμησης της εφαρμογής μας ανά υποστηριζόμενο είδος διαγράμματος.

1. Class Diagram (Διάγραμμα Κλάσεων)

Φυσική Περιγραφή (Prompt):
«Ένα σύστημα διαχείρισης UML διαγραμμάτων αποτελείται από χρήστες, έργα, διαγράμματα και σχόλια. Κάθε χρήστης έχει όνομα, email, κωδικό και ρόλο (user ή

admin). Κάθε χρήστης μπορεί να έχει πολλά έργα. Κάθε έργο έχει όνομα, περιγραφή και περιλαμβάνει πολλά διαγράμματα. Κάθε διάγραμμα έχει τύπο (class, sequence, activity, use case), περιγραφή σε φυσική γλώσσα, περιεχόμενο σε μορφή PlantUML και μια προκαθορισμένη μορφή εξαγωγής (PNG, SVG, ASCII). Επιπλέον, κάθε διάγραμμα μπορεί να έχει πολλά σχόλια, καθένα από τα οποία περιλαμβάνει κείμενο σχολίου και βαθμολογία. Οι ρόλοι καθορίζουν τα δικαιώματα του χρήστη. Ο admin μπορεί να διαχειρίζεται λογαριασμούς χρηστών, ενώ ο απλός χρήστης μόνο τα έργα και τα διαγράμματά του.»

Τύπος Διαγράμματος: Class Diagram
LLM: Gemini, **Μέγεθος**: Μικρό

Αξιολόγηση:

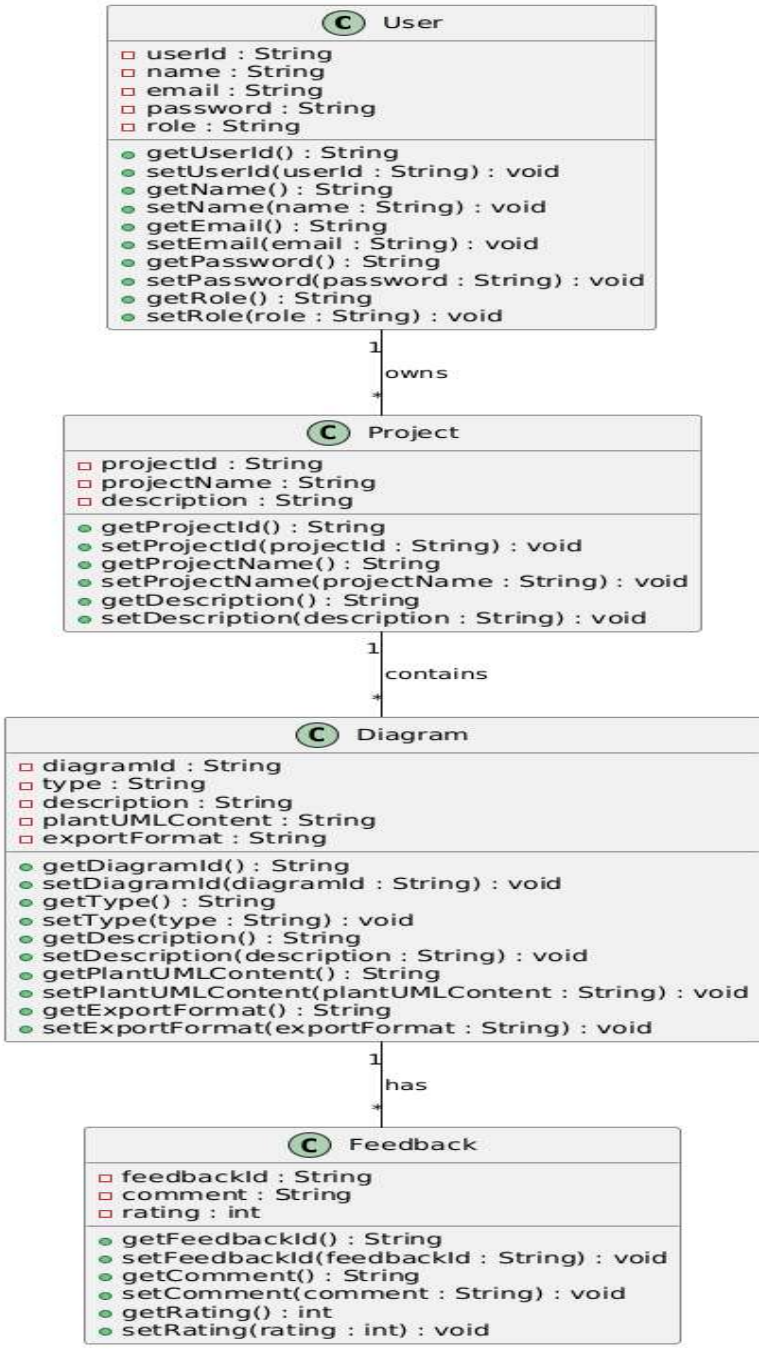
Πίνακας 4. Μετρικές αξιολόγησης για το διάγραμμα κλάσεων

Μετρική	*Τιμή*	*Παρατήρηση / Σχόλιο*
Συντακτική Ορθότητα	*100%*	*Το PlantUML script είναι συντακτικά έγκυρο και πλήρως συμβατό με την UML δομή.*
Ακρίβεια (Precision)	*100%*	*Όλες οι οντότητες, σχέσεις και πεδία που περιγράφηκαν στο σενάριο αποτυπώθηκαν σωστά.*
Ανάκληση (Recall)	*92%*	*Μικρή απόκλιση (~8%) λόγω απλής απόδοσης του role ως πεδίου αντί για υποκλάση.*
Σημασιολογική Ορθότητα (F1)	*96%*	*F1 ≈ 2 × (1 × 0.92) / (1 + 0.92) = 96%. Υψηλή συμφωνία με το gold standard.*
Χρόνος Εκτέλεσης	*1,91 δευτ.*	*Η παραγωγή ολοκληρώθηκε ταχύτερα από το όριο των δευτερολέπτων.*

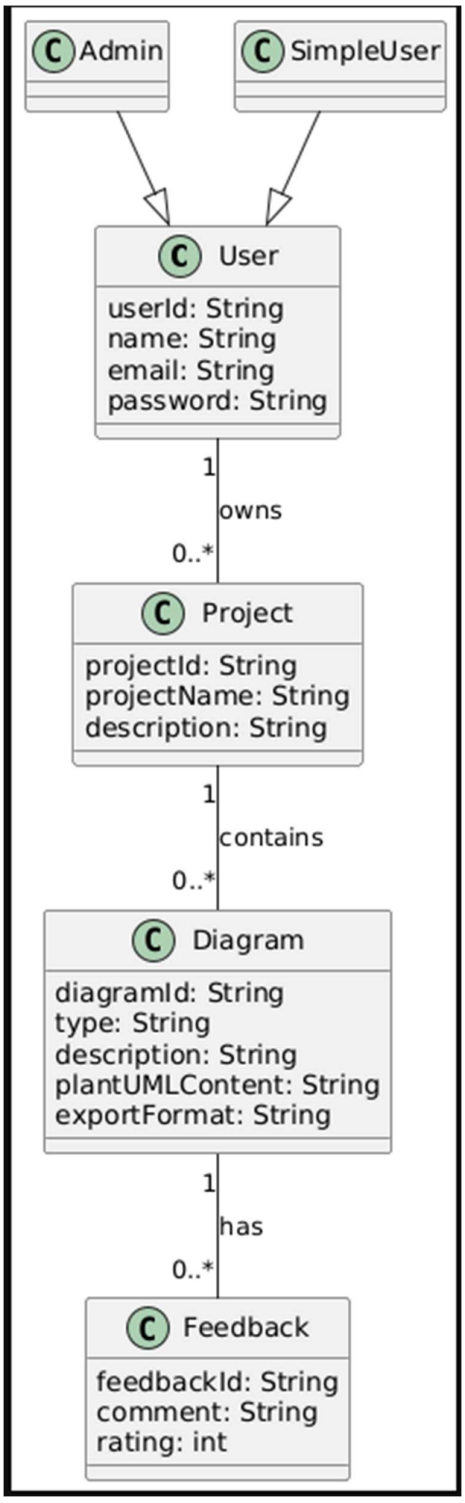
Συμπεράσματα:

Το παραγόμενο διάγραμμα (δείτε Εικόνα 34) **παρουσιάζει πολύ υψηλή συμφωνία** με το gold standard (δείτε Εικόνα 35). Οι κλάσεις, οι συσχετίσεις και τα πεδία έχουν αποδοθεί **πλήρως και σωστά**, με μόνη διαφοροποίηση τη μοντελοποίηση του *role* ως πεδίου αντί για υποκλάση. Αυτό δεν αναιρεί τη σημασιολογία, αλλά σε αυστηρή UML ανάλυση θεωρείται **απώλεια σχέσεων γενίκευσης**, μειώνοντας ελαφρώς την ανάκληση. Έτσι, η συνολική σημασιολογική ορθότητα (F1) κυμαίνεται στο **~96%**, επιβεβαιώνοντας ότι το σύστημα αποδίδει με μεγάλη ακρίβεια και πληρότητα.

Ο χρόνος απόκρισης ήταν **πολύ χαμηλός** (1,91 δευτ.), αποδεικνύοντας ότι η εφαρμογή ανταποκρίνεται σε **πραγματικό χρόνο**. Συνολικά, τα αποτελέσματα τεκμηριώνουν ότι το σύστημα καλύπτει **τόσο τη συντακτική εγκυρότητα όσο και τη σημασιολογική ακρίβεια**, ικανοποιώντας τους στόχους της αποτίμησης.


User
- userId : String
- name : String
- email : String
- password : String
- role : String
+ getUserId() : String
+ setUserId(userId : String) : void
+ getName() : String
+ setName(name : String) : void
+ getEmail() : String
+ setEmail(email : String) : void
+ getPassword() : String
+ setPassword(password : String) : void
+ getRole() : String
+ setRole(role : String) : void
1
owns
*
Project
- projectId : String
- projectName : String
- description : String
+ getProjectId() : String
+ setProjectId(projectId : String) : void
+ getProjectName() : String
+ setProjectName(projectName : String) : void
+ getDescription() : String
+ setDescription(description : String) : void
1
contains
*
Diagram
- diagramId : String
- type : String
- description : String
- plantUMLContent : String
- exportFormat : String
+ getDiagramId() : String
+ setDiagramId(diagramId : String) : void
+ getType() : String
+ setType(type : String) : void
+ getDescription() : String
+ setDescription(description : String) : void
+ getPlantUMLContent() : String
+ setPlantUMLContent(plantUMLContent : String) : void
+ getExportFormat() : String
+ setExportFormat(exportFormat : String) : void
1
has
*
Feedback
- feedbackId : String
- comment : String
- rating : int
+ getFeedbackId() : String
+ setFeedbackId(feedbackId : String) : void
+ getComment() : String
+ setComment(comment : String) : void
+ getRating() : int
+ setRating(rating : int) : void

Εικόνα 38. Το διάγραμμα κλάσεων που παράχθηκε από το LLM/σύστημα



Εικόνα 39. Το gold standard διάγραμμα κλάσεων

2. Sequence Diagram (Διάγραμμα Ακολουθίας)

Φυσική Περιγραφή (Prompt):
«Ο χρήστης εισάγει μια περιγραφή προβλήματος και επιλέγει τύπο διαγράμματος, μοντέλο LLM και μέγεθος. Αυτά στέλνονται στο backend της εφαρμογής. Το backend δημιουργεί ένα prompt και το αποστέλλει στο επιλεγμένο LLM. Το LLM επιστρέφει ένα UML διάγραμμα σε μορφή PlantUML. Το backend ελέγχει την εγκυρότητα του κώδικα. Αν είναι έγκυρος, το backend τον στέλνει στον renderer και επιστρέφει την εικόνα στον χρήστη. Αν υπάρχουν σφάλματα, ο χρήστης ενημερώνεται. Ο χρήστης έχει επίσης τη δυνατότητα να επεξεργαστεί απευθείας τον PlantUML κώδικα για βελτίωση του διαγράμματος χωρίς να χρειαστεί νέα υποβολή»

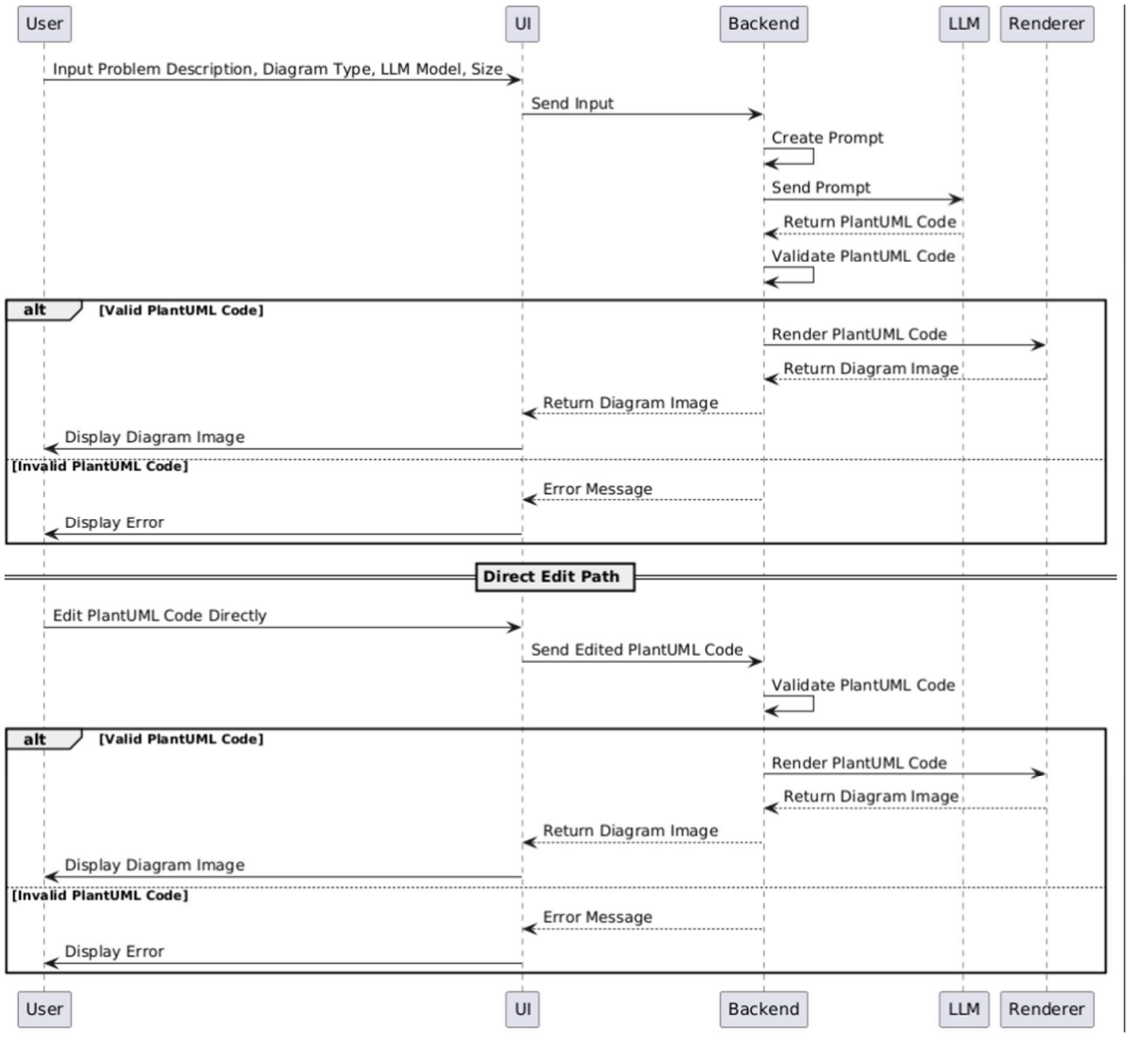
Τύπος Διαγράμματος: Sequence Diagram

LLM: Gemini
Μέγεθος: Μεγάλο

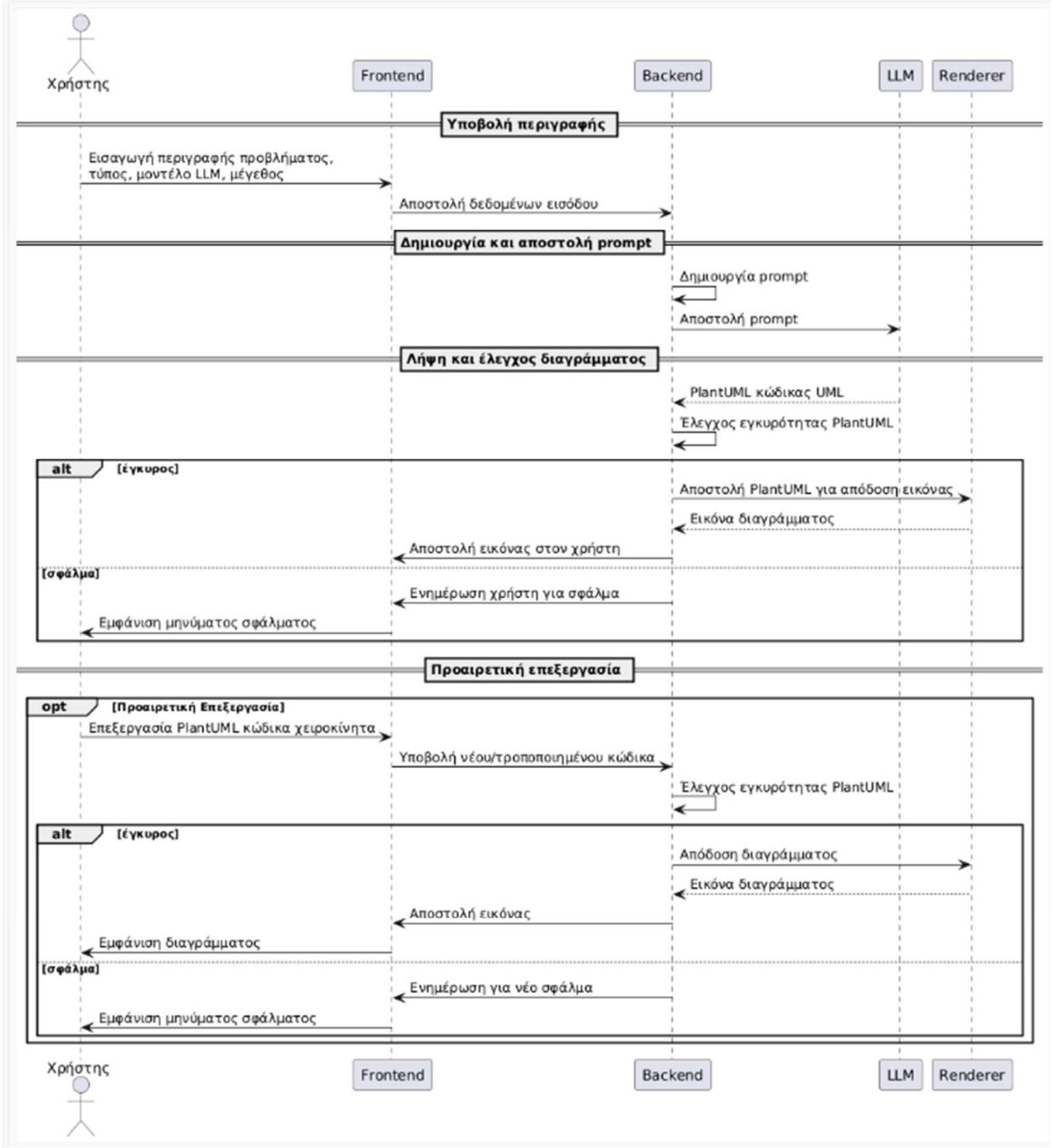
Αξιολόγηση:

Πίνακας 5. Μετρικές αξιολόγησης για το διάγραμμα ακολουθίας

Μετρική	***Τιμή***	***Παρατήρηση / Σχόλιο***
Συντακτική Ορθότητα	*100%*	*Το παραγόμενο διάγραμμα σε PlantUML είναι πλήρως συντακτικά έγκυρο.*
Precision (Ακρίβεια)	*100%*	*Όλες οι ενέργειες που αποτυπώνονται αντιστοιχούν σε ουσιαστικά στοιχεία του σεναρίου.*
Recall (Ανάκληση)	*96%*	Το *διάγραμμα δεν μοντελοποεί το τελευταίο μέρος της ροής ως προαιρετικό*
F1-score	*98%*	*Υψηλή συνολική σημασιολογική απόδοση με μικρές αποκλίσεις (F1 = 2×(1×0.96)/(1+0.96)).*
Χρόνος Εκτέλεσης	*1,57 sec*	*Η απόδοση του διαγράμματος πραγματοποιείται εντός των ορίων real-time λειτουργίας.*



Εικόνα 40. Το παραγόμενο διάγραμμα ακολουθίας από το LLM/σύστημα



Εικόνα 41. Το gold standard διάγραμμα ακολουθίας

Συμπεράσματα

Η αξιολόγηση του παραγόμενου διαγράμματος ακολουθίας έδειξε **πολύ υψηλή ακρίβεια και συντακτική εγκυρότητα**, γεγονός που επιβεβαιώνει την **ικανότητα του συστήματος να παράγει λειτουργικά και σωστά δομημένα UML διαγράμματα**. Το precision και η syntactic validity ανήλθαν στο **100%**,

αποδεικνύοντας ότι **το διάγραμμα περιέχει μόνο ουσιώδη και τεχνικά ορθά στοιχεία**.

Παράλληλα, η ανάκληση (Recall) παρουσίασε μικρή απόκλιση (**96%**), κυρίως λόγω **απουσίας της προαιρετικότητας (Opt) του τελευταίου μέρους της ροής**. Ωστόσο, η **συνολική σημασιολογική απόδοση** παραμένει αρκετά υψηλή (F1 ≈ **98%**), στοιχείο που υποδηλώνει **σαφήνεια και πληρότητα της κύριας ροής της διαδικασίας**.

Τέλος, ο χρόνος εκτέλεσης (1,57 δευτερόλεπτα) δείχνει ότι το σύστημα **ανταποκρίνεται άμεσα**, καλύπτοντας τις απαιτήσεις real-time αλληλεπίδρασης.

Συνολικά, το σύστημα αποδίδει με **ακρίβεια, αξιοπιστία και ταχύτητα**, με τις μικρές αποκλίσεις να εντοπίζονται σε **δευτερεύουσες φάσεις της διαδικασίας**, οι οποίες μπορούν να καλυφθούν σε μελλοντικές βελτιώσεις.

3.Activity Diagram (Διάγραμμα Δραστηριοτήτων)

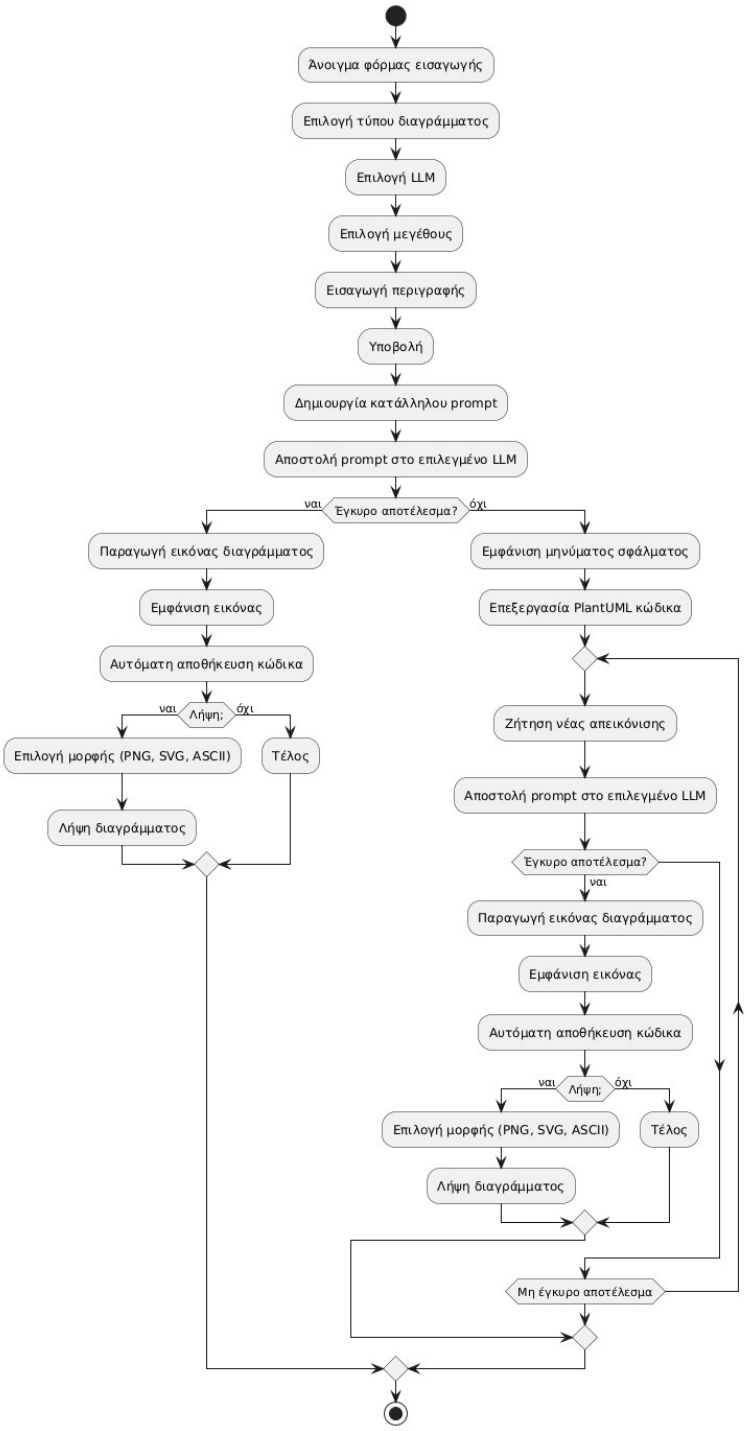
Φυσική Περιγραφή (Prompt):
«Ο χρήστης ανοίγει τη φόρμα εισαγωγής, επιλέγει τύπο διαγράμματος, LLM και μέγεθος, πληκτρολογεί περιγραφή και υποβάλλει. Το σύστημα δημιουργεί κατάλληλο prompt και το αποστέλλει στο επιλεγμένο LLM. Αν το αποτέλεσμα είναι έγκυρο, παράγεται και εμφανίζεται εικόνα του διαγράμματος. Αν το αποτέλεσμα περιέχει σφάλματα, εμφανίζεται σχετικό μήνυμα και ο χρήστης έχει τη δυνατότητα να επεξεργαστεί τον PlantUML κώδικα και να ζητήσει νέα απεικόνιση. Ο τελικός κώδικας διαγράμματος αποθηκεύεται αυτόματα από το σύστημα για μελλοντική χρήση ή τροποποίηση, ενώ ο χρήστης μπορεί προαιρετικά να προβεί σε λήψη (download) του διαγράμματος σε επιλεγμένη μορφή (PNG, SVG ή ASCII).

Τύπος Διαγράμματος: Activity Diagram

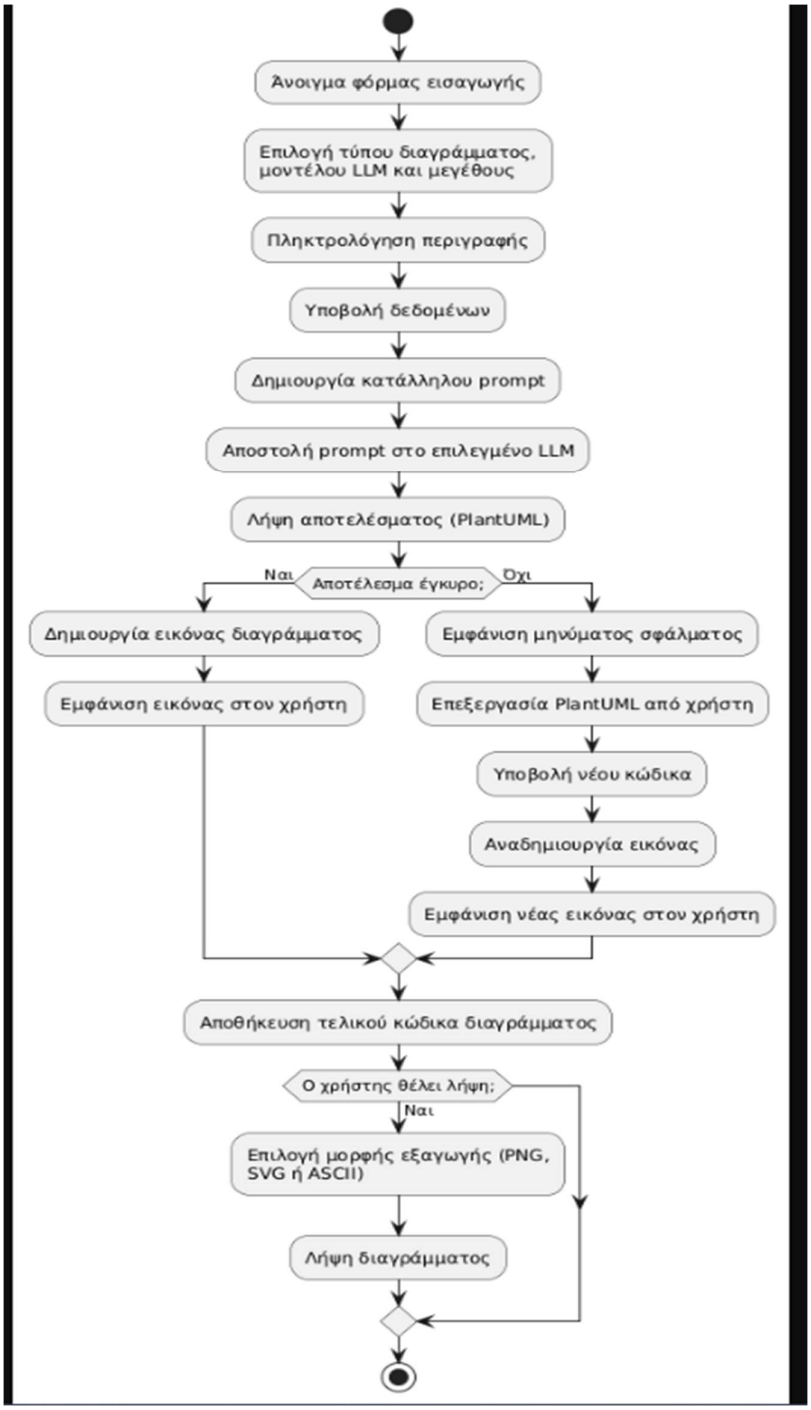
LLM: Gemini, **Μέγεθος**: Μικρό

Πίνακας 6. Μετρικές αξιολόγησης για το διάγραμμα δραστηριοτήτων

Μετρική	***Τιμή***	***Παρατήρηση / Σχόλιο***
Συντακτική Ορθότητα	*100%*	*Δεν υπάρχουν συντακτικά σφάλματα — το διάγραμμα αποδίδεται κανονικά.*
Precision	*88%*	*Περιλαμβάνει λανθασμένο μονοπάτι που ξανακαλεί το LLM σε περιπτώσεις σφάλματος κώδικα. Περιλαλαμβάνει αποφάσεις με μια μόνο εξερχόμενη ροή χωρίς ετικέτα και μια δραστηριότητα χωρίς νόημα*
Recall	*85%*	*Δεν αποτυπώνεται η σωστή ροή επεξεργασίας τροποποιημένου PlantUML κώδικα.*
F1-score	*86.5%*	*Μέτρια συμφωνία με το gold standard λόγω εννοιολογικών αποκλίσεων.*
Χρόνος Εκτέλεσης	*1,65*	*Εντός ορίων real-time — καμία υστέρηση στην απεικόνιση.*



Εικόνα 42. Το παραγόμενο διάγραμμα δραστηριοτήτων από το LLM/σύστημα



Εικόνα 43. Το gold standard διάγραμμα δραστηριοτήτων

Συμπεράσματα

Η αξιολόγηση του παραγόμενου διαγράμματος ανέδειξε **υψηλή συντακτική εγκυρότητα** και **ικανοποιητική σημασιολογική συμφωνία** με το gold standard. Το διάγραμμα αποτυπώνει με ακρίβεια τη βασική ροή της διαδικασίας εισαγωγής, επεξεργασίας και απόδοσης UML διαγραμμάτων, καθώς και τη διαχείριση σφαλμάτων.

Ωστόσο, εντοπίστηκε **δομική απόκλιση στην υπο-ροή που ακολουθείται όταν ο κώδικας δεν είναι έγκυρος**. Συγκεκριμένα, το σύστημα εμφανίζεται να αποστέλλει νέο prompt στο LLM, πράγμα που είναι εννοιολογικά λανθασμένο, καθώς η σωστή ενέργεια είναι ο **έλεγχος του τροποποιημένου PlantUML κώδικα** — χωρίς εμπλοκή του LLM. Αυτή η απόκλιση επηρεάζει αρνητικά τόσο την ακρίβεια (precision) όσο και την πληρότητα (recall) του διαγράμματος. Επιπλέον, εμφανίζει και άλλα σημασιολογικά λάθη, όπως αποφάσεις (πχ. Μη Έγκυρο Αποτέλεσμα) χωρίς υπο-ροές με ταμπέλες καθώς και δραστηριότητες χωρίς νόημα (πχ. Τέλος).

Παρά την απόκλιση, η **F1-score διατηρείται σε υψηλά επίπεδα**, υποδεικνύοντας πως ο πυρήνας της επιχειρησιακής λογικής έχει αποδοθεί με σχετική επιτυχία. Ο **χρόνος εκτέλεσης** ήταν επίσης εξαιρετικός, επιβεβαιώνοντας την αποδοτικότητα της παραγωγής.

4. Use Case Diagram (Διάγραμμα Περιπτώσεων Χρήσης)

Φυσική Περιγραφή (Prompt):

*«Το σύστημα διαχείρισης UML διαγραμμάτων υποστηρίζει τρεις τύπους χρηστών: **SimpleUser**, **Administrator** και **AdvancedUser**, με σχέσεις γενίκευσης μεταξύ τους.*

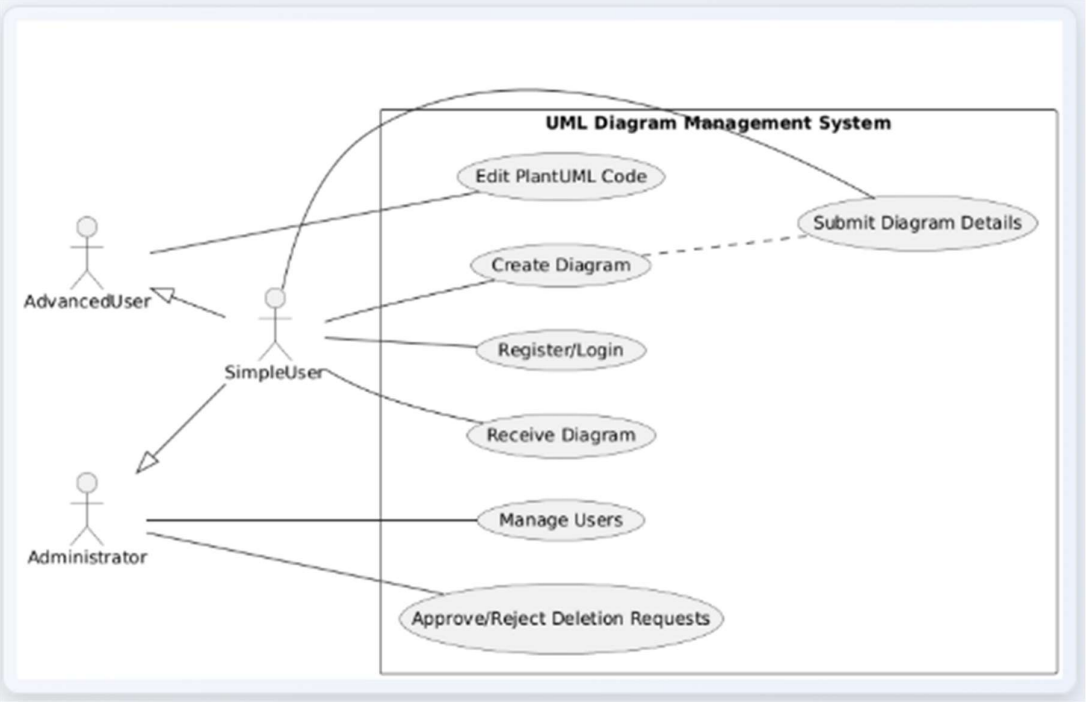
- *Ο **SimpleUser** μπορεί να:*
  - *Εγγράφεται και να συνδέεται στο σύστημα.*
  - *Δημιουργεί ένα νέο διάγραμμα, υποβάλλοντας τα απαραίτητα στοιχεία (τύπο, μοντέλο LLM, περιγραφή).*
  - *Λαμβάνει το διάγραμμα που έχει παραχθεί, εφόσον το επιθυμεί.*
- *Ο **Administrator** μπορεί επιπλέον:*
  - *Να διαχειρίζεται χρήστες.*
  - *Να εγκρίνει ή να απορρίπτει αιτήσεις διαγραφής λογαριασμών.*
- *Ο **AdvancedUser** μπορεί:*
  - *Να επεξεργάζεται τον PlantUML κώδικα που παράγεται από το σύστημα.*

*Η υποβολή των στοιχείων είναι μέρος της διαδικασίας δημιουργίας διαγράμματος. Η λήψη του διαγράμματος μπορεί να γίνει **μόνο αν έχει ήδη παραχθεί** και ο χρήστης το επιθυμεί (προαιρετικό βήμα).»*

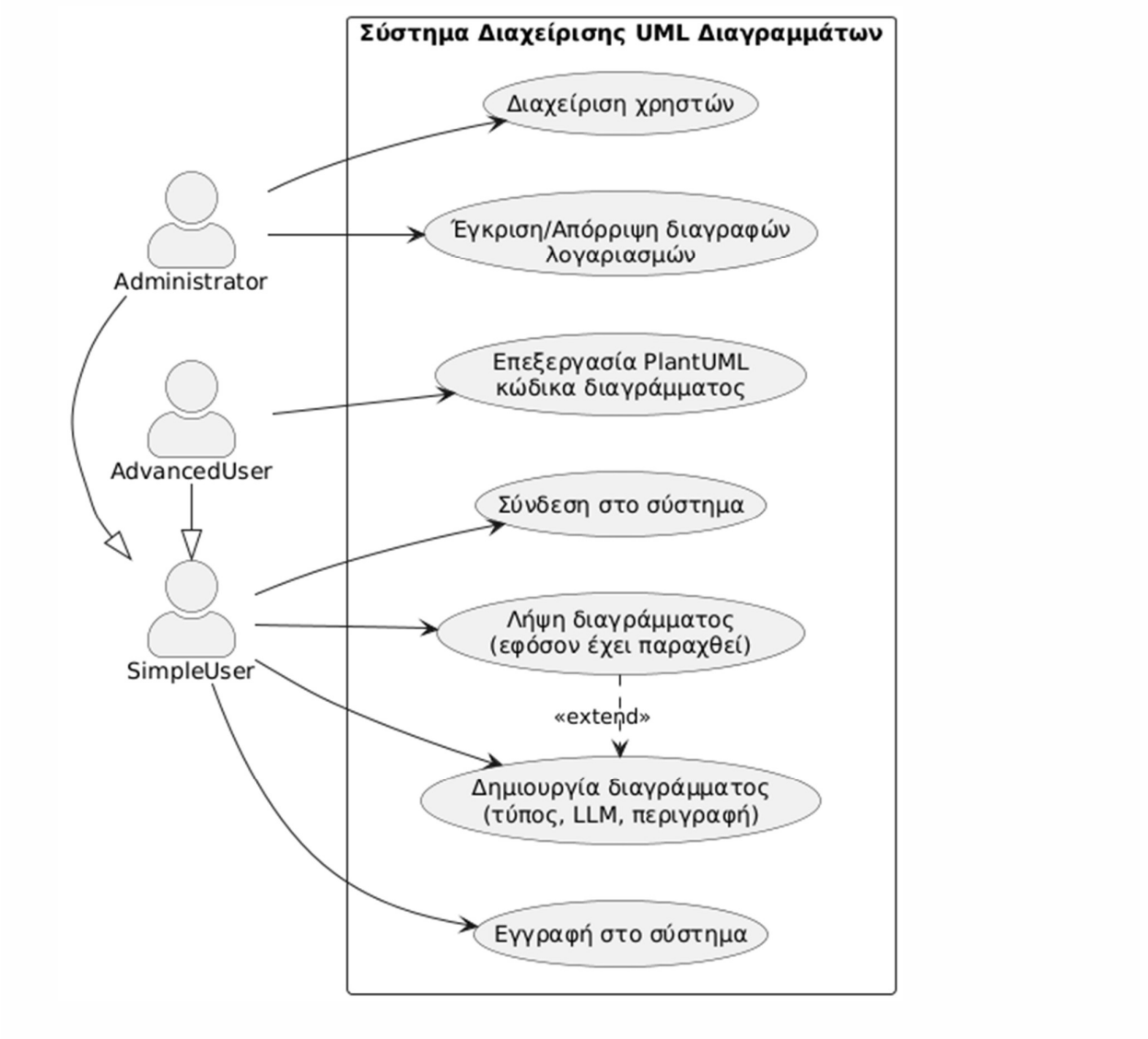
Τύπος Διαγράμματος: Use Case Diagram
LLM: GPT 4, **Μέγεθος**: Μικρό

Πίνακας 7. Μετρικές αξιολόγησης για το διάγραμμα περιπτώσεων χρήσης

Μετρική	Τιμή	Σχόλιο
Syntactic Validity	100%	Ορθές UML συνδέσεις & συμβολισμοί
Precision	83%	Πλεονάζουσα περίπτωση χρήσης + λάθος ISA
Recall	75%	Απουσία σχέσεων <<extend>> & διαχωρισμού login/register
F1-score	78.8%	F1 = 2 × (0.83 × 0.75) / (0.83 + 0.75)
Execution Time	1.43 sec	Εντός των real-time ορίων



Εικόνα 44. Το διάγραμμα περιπτώσεων χρήσης που παράγεται από το LLM/σύστημα



Εικόνα 45. Το gold standard διάγραμμα περιπτώσεων χρήσης

Συμπεράσματα:

Η αξιολόγηση του παραγόμενου use case diagram ανέδειξε ότι το σύστημα έχει τη δυνατότητα να παραγάγει **συντακτικά άρτια και λογικά κατανοητά διαγράμματα**, τα οποία καλύπτουν βασικές λειτουργίες και χρήστες του συστήματος. Όλες οι συσχετίσεις μεταξύ actors και use cases είναι συμβατές με τη γλώσσα UML και αποδίδονται ορθά.

Ωστόσο, σε επίπεδο σημασιολογικής ορθότητας εντοπίστηκαν **σημαντικές αποκλίσεις**:

- Δεν περιλαμβάνονται δύο σχέσεις `<<extend>>` που είναι κρίσιμες για την κατανόηση της εξάρτησης βασικών λειτουργιών του συστήματος, όπως η **λήψη** και η **επεξεργασία** διαγραμμάτων από τη δημιουργία τους.
- Ο διαχωρισμός μεταξύ **εγγραφής** και **σύνδεσης** παρουσιάζεται ως ενιαία περίπτωση χρήσης (`Register/Login`), πράγμα που είναι λάθος.
- Υπάρχει ένα πλεονάζον use case ("Submit Diagram Details") που δεν υφίσταται στο gold standard και οι ISA σχέσεις έχουν αποδοθεί ανάποδα!
- Οι ISA σχέσεις έχουν ανάποδη φορά

.

Παρά τα παραπάνω, η συνολική **F1-score (78.8%)** δείχνει ότι **ο πυρήνας της λειτουργικότητας αποδίδεται ικανοποιητικά** ενώ και η **χρονική απόδοση** (1,43 sec) είναι **άριστη**, υποδεικνύοντας γρήγορη επεξεργασία και απόδοση του διαγράμματος.

6.3 Συνολικά Συμπεράσματα Αξιολόγησης

Η συνολική αξιολόγηση της λειτουργικότητας της εφαρμογής, όπως προέκυψε από τη σύγκριση παραγόμενων UML διαγραμμάτων με αντίστοιχα gold standard διαγράμματα, καταδεικνύει την **ικανότητα του συστήματος να αποδίδει αξιόπιστα, συντακτικά ορθά και ως επί τω πλείστο σημασιολογικά συνεπή μοντέλα**.

Παρά τις επιμέρους αποκλίσεις σε συγκεκριμένα διαγράμματα (όπως παραλείψεις σχέσεων γενίκευσης ή <<extend>>, ή λανθασμένη απόδοση της ροής σε περίπτωση σφάλματος), η εφαρμογή διατηρεί **σταθερά υψηλές επιδόσεις** σε όλες τις βασικές μετρικές αποτίμησης.

Πίνακας 8. Συνολικά Συμπεράσματα Αξιολόγησης

Μετρική	**Μέση Τιμή**
Συντακτική Ορθότητα	100%
Precision (Ακρίβεια)	~92.75%
Recall (Ανάκληση)	~87%
F1-score	~89.5%
Χρόνος Εκτέλεσης	~1.64 sec

Τα παραπάνω ποσοστά επιβεβαιώνουν ότι το σύστημα αποδίδει **συντακτικά πλήρη** διαγράμματα, με **υψηλό επίπεδο σημασιολογικής ακρίβειας και πληρότητας**, ακόμα και σε σενάρια με πιο σύνθετη επιχειρησιακή λογική (π.χ. sequence ή activity diagrams).

Ειδικότερα:

- Η **συντακτική εγκυρότητα** είναι σταθερά 100% σε όλα τα διαγράμματα, υποδεικνύοντας την αξιόπιστη παραγωγή έγκυρου UML κώδικα.
- Η **ακρίβεια (Precision)** κυμαίνεται πολύ ψηλά (~93%), γεγονός που δείχνει ότι το σύστημα σπάνια εισάγει άσχετες ή εσφαλμένες πληροφορίες.
- Η **ανάκληση (Recall)** επηρεάζεται κυρίως από παραλείψεις υπο-ροών ή σχέσεων εξάρτησης (extend, ISA), χωρίς όμως να αλλοιώνεται ο βασικός πυρήνας των διαγραμμάτων.
- Ο μέσος **χρόνος εκτέλεσης (~1,64 δευτ.)** κατατάσσει την εφαρμογή στα συστήματα real-time, κατάλληλα για άμεση ανατροφοδότηση σε διαδραστικά περιβάλλοντα.

Συμπερασματικά, η εφαρμογή ικανοποιεί τους στόχους σχεδίασης, παρέχοντας **αποτελέσματα υψηλής ποιότητας**, με **πλήρη συντακτική συμβατότητα**, **ικανοποιητική σημασιολογική συμφωνία** και **πολύ καλή χρονική απόδοση**. Οι επισημασμένες αποκλίσεις μπορούν να καλυφθούν μέσω εστιασμένων βελτιώσεων στις ροές ελέγχου και στον χειρισμό ειδικών περιπτώσεων (ISA, extend), γεγονός που υποδηλώνει και τις **προοπτικές μελλοντικής εξέλιξης του συστήματος**.

Κεφάλαιο 7 – Συμπεράσματα και Μελλοντική Εργασία

7.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία είχε ως βασικό στόχο την ανάπτυξη μιας πλήρους και λειτουργικής εφαρμογής ιστού που αξιοποιεί τις δυνατότητες των Μεγάλων Γλωσσικών Μοντέλων (LLMs) για την αυτόματη παραγωγή UML διαγραμμάτων από περιγραφές χρηστών σε φυσική γλώσσα. Το τελικό προϊόν επιβεβαιώνει την ισχύ και τη χρησιμότητα των σύγχρονων εργαλείων Τεχνητής Νοημοσύνης στον χώρο της μοντελοποίησης συστημάτων λογισμικού, συνδυάζοντας καινοτομία, χρηστικότητα και τεχνική επάρκεια.

Η βασική συνεισφορά της εργασίας έγκειται στη σύνθεση διαφορετικών τεχνολογιών σε ένα ενιαίο και εύχρηστο περιβάλλον: το σύστημα επιτρέπει στον χρήστη να δημιουργεί UML διαγράμματα χωρίς να απαιτείται γνώση εξειδικευμένων συμβολισμών ή εργαλείων. Η δυνατότητα επιλογής μεταξύ διαφορετικών παρόχων LLMs, όπως GPT, Gemini και DeepSeek, ενισχύει την ευελιξία και τη συγκριτική αξιολόγηση των αποτελεσμάτων. Παράλληλα, η υποστήριξη πολλών τύπων διαγραμμάτων, η δυνατότητα εξαγωγής τους σε διάφορες μορφές, η και χρηστών καθιστούν την εφαρμογή κατάλληλη για χρήση σε πραγματικά περιβάλλοντα ανάπτυξης.

Μέσα από τη διαδικασία σχεδίασης, υλοποίησης και τεκμηρίωσης του συστήματος, αποκτήθηκε σε βάθος εμπειρία στον σχεδιασμό εφαρμογών ιστού που βασίζονται σε τεχνολογίες Τεχνητής Νοημοσύνης. Επιπλέον, ενισχύθηκε η εξοικείωση με τις έννοιες του prompt engineering, της RESTful σχεδίασης API, της ασφάλειας δεδομένων, αλλά και της επικοινωνίας με πολυσύνθετα εξωτερικά συστήματα, όπως τα LLMs. Το έργο αυτό ανέδειξε τη δυνατότητα ουσιαστικής σύνδεσης μεταξύ της φυσικής γλώσσας και της τυπικής αναπαράστασης λογισμικού, προσφέροντας μια σύγχρονη προσέγγιση στην παραγωγή σχεδιαστικών μοντέλων.

7.2 Μελλοντική Εργασία

Παρότι η προτεινόμενη εφαρμογή ιστού καλύπτει ένα ευρύ φάσμα λειτουργιών, υπάρχουν αρκετά σημεία στα οποία μπορεί να επεκταθεί στο μέλλον, ενισχύοντας τόσο τη λειτουργικότητά της όσο και τη συμβολή της στην έρευνα και την πράξη.

Μια πρώτη κατεύθυνση αφορά την ενσωμάτωση **μηχανισμών αυτόματης σύγκρισης αποτελεσμάτων** μεταξύ διαφορετικών LLMs, με χρήση μετρικών, όπως η **ακρίβεια**, η **πληρότητα** και η **συντακτική ορθότητα** του παραγόμενου μοντέλου PlantUML. Μέσω αυτής της δυνατότητας, ο χρήστης θα μπορεί να επιλέγει το καταλληλότερο μοντέλο για κάθε είσοδο, λαμβάνοντας υπόψη ποιοτικά και ποσοτικά κριτήρια. Ωστόσο, για να είναι η σύγκριση αυτή **αντικειμενική, επαναλήψιμη και αυτοματοποιημένη**, απαιτείται η δημιουργία κατάλληλου **συνόλου δεδομένων αξιολόγησης (evaluation dataset)**. Το σύνολο αυτό θα περιλαμβάνει περιγραφές σε φυσική γλώσσα (prompts) συνοδευόμενες από **προκαθορισμένα UML διαγράμματα-στόχους (ground truth models)**. Αυτά θα χρησιμοποιούνται ως

σημεία αναφοράς για τον υπολογισμό των μετρικών απόδοσης, μέσω συστηματικής αντιπαραβολής με τις εξόδους των LLMs. Η προσέγγιση αυτή προϋποθέτει όχι μόνο την τεχνική επέκταση της εφαρμογής ώστε να υποστηρίζει πολλαπλά μοντέλα και υπολογισμό μετρικών, αλλά και τη σχεδίαση ενός **αντιπροσωπευτικού συνόδου δεδομένων**, κατάλληλου για συγκριτική αποτίμηση.

Μια δεύτερη κατεύθυνση αφορά την ενσωμάτωση δυνατότητας εκπαίδευσης προσαρμοσμένων LLMs (custom LLMs) σε συγκεκριμένους θεματικούς τομείς, όπως η εκπαίδευση, η ιατρική ή η νομική πληροφορική. Η λειτουργία αυτή μπορεί να υλοποιηθεί μέσω τεχνικών fine-tuning, αξιοποιώντας εξειδικευμένα σύνολα δεδομένων ανά τομέα (domain), ώστε το μοντέλο να μάθει τη σχετική ορολογία, τη λογική δομή και τις απαιτήσεις του κάθε πεδίου. Εφαρμοσμένες τεχνικές fine-tuning περιλαμβάνουν μεθόδους, όπως το Low-Rank Adaptation (LoRA) (Hu et al., 2021) ή άλλες παραμετροποιήσιμες προσεγγίσεις (Parameter-Efficient Fine-Tuning – PEFT) (Dettmers et al., 2022), οι οποίες επιτρέπουν την αποδοτική εκπαίδευση μοντέλων σε περιορισμένα ή εξειδικευμένα σύνολα δεδομένων..

Από την αξιολόγηση του συστήματος προκύπτει ότι, σε πιο σύνθετες ή ορολογικά απαιτητικές περιπτώσεις, τα LLMs δεν αποδίδουν πάντα με ακρίβεια, κυρίως επειδή τα σύνολα δεδομένων εκπαίδευσής τους δεν περιλαμβάνουν επαρκή ποσότητα και ποικιλία για εξειδικευμένες περιοχές. Για τον λόγο αυτό, η δυνατότητα fine-tuning εμφανίζεται κρίσιμη για την παραγωγή πιο ακριβών, συνεπών και σημασιολογικά ορθών UML διαγραμμάτων, ιδιαίτερα σε εφαρμογές που απαιτούν υψηλή αξιοπιστία.

Η κατεύθυνση αυτή μπορεί να ενισχυθεί περαιτέρω **σε συνδυασμό με την πρώτη κατεύθυνση**, μέσω της δημιουργίας **domain-specific benchmark sets**, τα οποία βασίζονται σε ειδικά ως προς το πεδίο εφαρμογής δεδομένα και επιτρέπουν την αποτίμηση των fine-tuned μοντέλων και τη σύγκρισή τους με γενικά LLMs ως προς την απόδοση στα συγκεκριμένα πεδία εφαρμογής που καλύπτονται από τα benchmark sets.

Η υποστήριξη για εισαγωγή πηγαίου κώδικα (π.χ. Python, Java) και την αυτόματη παραγωγή UML διαγραμμάτων μέσω **στατικής ανάλυσης (static analysis)** ή με χρήση LLMs (code-to-UML), αποτελεί μια σημαντική μελλοντική δυνατότητα της εφαρμογής. Το σενάριο αυτό αποκτά ιδιαίτερη σημασία στο πλαίσιο του **reverse engineering**, δηλαδή της ανακατασκευής δομικών μοντέλων από υπάρχοντα συστήματα. Ωστόσο, η παραγωγή UML μοντέλων από πηγαίο κώδικα θα πρέπει να γίνεται με **ευφυή τρόπο**, ανάλογα με τον τύπο του διαγράμματος προς δημιουργία (π.χ. κλάσεων, ακολουθίας κ.λπ.). Συχνά, για την παραγωγή ενός μόνο διαγράμματος απαιτείται η ανάλυση **πολλαπλών αρχείων κώδικα**, γεγονός που μπορεί να ξεπερνά το όριο εισόδου (context length) που υποστηρίζει το κάθε LLM. Τα όρια αυτά διαφέρουν ανάλογα με το μοντέλο και πρέπει να λαμβάνονται υπόψη κατά τον σχεδιασμό της ροής. Αντίστροφα, το **σενάριο UML-to-code** παρουσιάζει ιδιαίτερο ενδιαφέρον, καθώς επιτρέπει στον χρήστη να δημιουργεί ή να τροποποιεί ένα διάγραμμα και να λαμβάνει αυτόματα τον αντίστοιχο πηγαίο κώδικα σε γλώσσα επιλογής (π.χ. Python). Ακόμα και αν ο παραγόμενος κώδικας απαιτεί ορισμένες βελτιώσεις, αποτελεί μια **γρήγορη και εύχρηστη αφετηρία** ανάπτυξης. Αυτό εξυπηρετεί ιδιαίτερα περιβάλλοντα όπου η αρχική μοντελοποίηση είναι εκτενής και το ζητούμενο είναι η **ημι-αυτόματη παραγωγή λειτουργικού κώδικα** από μοντέλα (σενάρια model-driven engineering). Η δυνατότητα οπτικής τροποποίησης του

διαγράμματος από τον χρήστη (graphical editing) και αυτόματης μετατροπής των αλλαγών σε PlantUML ενισχύει ακόμη περισσότερο αυτή την αμφίδρομη ροή.

Η εφαρμογή μπορεί να επεκταθεί ώστε να υποστηρίζει **πολύγλωσση διεπαφή** (π.χ. Ελληνικά, Αγγλικά), τόσο σε επίπεδο γραφικών στοιχείων (UI), όσο και στο επίπεδο της εισόδου του χρήστη, με την υποστήριξη **πολύγλωσσων prompts** και αυτόματης μετάφρασης. Αυτό καθιστά την εφαρμογή πιο προσβάσιμη και κατάλληλη για χρήση σε **διεθνή ή εκπαιδευτικά περιβάλλοντα**. Επιπλέον, μπορεί να προστεθεί δυνατότητα **εξαγωγής του παραγόμενου διαγράμματος σε PDF**, συνοδευόμενου από περιγραφική τεκμηρίωση. Το κείμενο της τεκμηρίωσης μπορεί:

- Είτε να βασίζεται **στην αρχική περιγραφή του χρήστη (prompt)**
- Είτε να **παράγεται αυτόματα από το LLM**, με φυσική γλώσσα, και να συνοψίζει το περιεχόμενο και τη δομή του παραχθέντος διαγράμματος π.χ.("Το παραγόμενο διάγραμμα περιλαμβάνει τρεις βασικέςκλάσεις...").

Η δεύτερη προσέγγιση προσφέρει επιπλέον **προστιθέμενη αξία**, καθώς αξιοποιεί τη φυσική γλωσσική ικανότητα του LLM για τη δημιουργία **τεχνικής τεκμηρίωσης**, η οποία μπορεί να ενσωματωθεί σε μεγαλύτερα έγγραφα, προδιαγραφές ή παρουσιάσεις.

Ένα στοιχείο που θα ενίσχυε περαιτέρω τη δυναμική του συστήματος που αναπτύχθηκε είναι η υποστήριξη **συνεργατικής επεξεργασίας (collaborative modeling)**, όπου πολλοί χρήστες μπορούν να επεξεργάζονται ταυτόχρονα το ίδιο διάγραμμα, είτε την αρχική περιγραφή, είτε το παραγόμενο μοντέλο (PlantUML), είτε το ίδιο το διάγραμμα σε γραφική μορφή, σε **πραγματικό χρόνο**. Αυτό θα πρέπει να συνδυαστεί με τον έλεγχο εκδόσεων διαγράμματος ώστε να μπορεί εύκολα να γίνεται η μετάβαση από την τρέχουσα σε παλαιότερη έκδοση του διαγράμματος εφόσον η τρέχουσα δεν είναι ικανοποιητική ή ενδεχομένως παρουσιάζει διάφορα προβλήματα (πχ. ποιότητας ή πληρότητας).

Σε προχωρημένο στάδιο, η εφαρμογή μπορεί να λειτουργήσει και ως **σύστημα αξιολόγησης UML γνώσεων**, μέσα από δημιουργία και αυτόματη διόρθωση ασκήσεων.

Η υλοποίηση ενός feedback loop με self-improving prompts, δηλαδή prompts που προσαρμόζονται βάσει της ανατροφοδότησης του χρήστη, μπορεί να προσδώσει σημαντική νοημοσύνη στο ίδιο το σύστημα παραγωγής. Ένα κρίσιμο σημείο για τη βελτίωση της απόδοσης της εφαρμογής στο μέλλον αφορά την **οργάνωση, καταγραφή και ανάλυση των prompts** που χρησιμοποιούνται για την παραγωγή κάθε τύπου διαγράμματος (Class, Sequence, Use Case, Activity). Στην παρούσα έκδοση του συστήματος, τα prompts παράγονται με **ημι-αυτόματο τρόπο**, με βάση την είσοδο του χρήστη και μια σειρά από έτοιμα templates, τα οποία **εμπλουτίζονται με τις επιλογές του χρήστη** (τύπος διαγράμματος, LLM, μέγεθος, κ.ά.). Ωστόσο, δεν έχει ακόμη υλοποιηθεί ένα **μηχανιστικό σύστημα καταγραφής και αποτίμησης** των prompts ανά τύπο διαγράμματος και παραγόμενο αποτέλεσμα. Η ύπαρξη μιας τέτοιας βάσης δεδομένων prompts θα επέτρεπε τη **συστηματική μελέτη της αποτελεσματικότητάς τους**, την αποτύπωση βέλτιστων πρακτικών και τελικά τη δημιουργία μιας **βιβλιοθήκης prompt templates**, τα οποία θα εξελίσσονται

δυναμικά. Η παραπάνω διαδικασία μπορεί να υποστηριχθεί περαιτέρω από την ενσωμάτωση **μηχανισμών AI explainability**. Για παράδειγμα, εργαλεία που αναλύουν και εξηγούν **"γιατί" προτάθηκε μια συγκεκριμένη δομή** (π.χ. σχέση συσχέτισης ή κληρονομικότητας), μπορούν να ενισχύσουν τόσο:

- Την **εμπιστοσύνη** του χρήστη στο παραγόμενο αποτέλεσμα,
- Όσο και τη **διδακτική αξία** της εφαρμογής (εκπαιδευτικό λογισμικό μοντελοποίησης).

Η **συστηματική βελτίωση των prompts**, ώστε να παράγονται **πιο ακριβή και σημασιολογικά ορθά μοντέλα** αποτελεί μια πολλά υποσχόμενη κατεύθυνση. Από την αξιολόγηση προέκυψε ότι τα LLMs τείνουν να υπερ-αναλύουν ή να παραλείπουν στοιχεία, όταν η περιγραφή δεν είναι αρκετά σαφής ή όταν τα prompts δεν εξειδικεύουν σωστά τις σχέσεις και τους περιορισμούς. Η ανάπτυξη **ειδικών βιβλιοθηκών prompts**, προσαρμοσμένων ανά τύπο διαγράμματος ή τομέα εφαρμογής, μπορεί να καλύψει αυτά τα κενά, όπως αναφέρθηκε και στην προηγούμενη παράγραφο. Αυτό συνδέεται κάθετα με την ανάγκη εξειδίκευσης σε τομείς, όπως η εκπαίδευση ή η νομική πληροφορική, για τους οποίους τα γενικά LLMs δεν επαρκούν από μόνα τους.

Μια τελευταία κατεύθυνση επέκτασης είναι η υλοποίηση **συστήματος διαχείρισης έργων μοντελοποίησης** (*project-based modeling*). Αυτό θα επιτρέπει στον χρήστη να οργανώνει πολλαπλά διαγράμματα ανά έργο, να τα ομαδοποιεί θεματικά ή χρονικά, να τα διαχειρίζεται σε επίπεδο εκδόσεων και να παρακολουθεί την πρόοδο σχεδίασης/μοντελοποίησής τους. Η δυνατότητα αυτή συνδέεται άμεσα με την ανάγκη οργάνωσης της εργασίας, ειδικά σε σύνθετα συστήματα όπου τα διαγράμματα UML αλληλοσυνδέονται και εξελίσσονται σταδιακά.

Κεφάλαιο 8 – Βιβλιογραφία

1. Booch, G., Rumbaugh, J. and Jacobson, I., 2005. *The Unified Modeling Language User Guide*. 2nd ed. Boston: Addison-Wesley.
2. Object Management Group (OMG), 2017. *OMG Unified Modeling Language (UML), Version 2.5.1*. [online] Διαθέσιμο στο: https://www.omg.org/spec/UML/2.5.1/ [Πρόσβαση 2 Φεβ. 2025].
3. Fowler, M., 2003. *UML distilled: a brief guide to the standard object modeling language*. 3rd ed. Boston: Addison-Wesley.
4. PlantUML, 2023. *PlantUML Language Reference Guide*. [online] Διαθέσιμο στο: https://plantuml.com/ [Πρόσβαση 2 Φεβ. 2025].
5. Lucid Software Inc., 2024. *Lucidchart – Online UML Diagram Tool*. [online] Διαθέσιμο στο: https://www.lucidchart.com/ [Πρόσβαση 9 Φεβ. 2025].
6. OpenAI, 2023. *GPT-4 Technical Report*. [online] Διαθέσιμο στο: https://openai.com/research [Πρόσβαση 27 Φεβ. 2025].
7. Google DeepMind, 2024. *Gemini 1.5 Technical Overview*. [online] Διαθέσιμο στο: https://deepmind.google/technologies/gemini [Πρόσβαση 27 Μαρ. 2025].
8. Gilbert, P., 2022. *The Role of Generative AI in Software Engineering*. *Software Engineering Journal*, 37(4), pp.245–258.
9. Ferrari, A., Abualhaija, S. και Arora, C., 2024. Model Generation with LLMs: From Requirements to UML Sequence Diagrams. *arXiv preprint*. Διαθέσιμο στο: https://arxiv.org/abs/2404.06371 [Πρόσβαση 18 Μαρ. 2025].
10. Conrardy, A. και Cabot, J., 2024. First Results of Image Based UML Diagram Generation Using LLMs. *arXiv preprint*. Διαθέσιμο στο: https://arxiv.org/abs/2404.11376 [Πρόσβαση 18 Μαρ. 2025].
11. Bates, A., Vavricka, R., Chen, H. και Sharma, R., 2025. Unified Modeling Language Code Generation from Diagram Images Using Multimodal Large Language Models. *arXiv preprint*. Διαθέσιμο στο: https://arxiv.org/abs/2503.12293 [Πρόσβαση 20 Μαρ. 2025].

12. DeepSeek, 2024. DeepSeek Developer Platform. [online] Διαθέσιμο στο: https://platform.deepseek.com/ [Πρόσβαση 28 Μαρ. 2025].

13. Google DeepMind, 2024. Gemini 1.5 Technical Overview. [online] Διαθέσιμο στο: https://deepmind.google/technologies/gemini [Πρόσβαση 27 Μαρ. 2025].

14. OpenAI, 2023. GPT-4 Technical Report. [online] Διαθέσιμο στο: https://openai.com/research/gpt-4 [Πρόσβαση 27 Φεβ. 2025].

15. Méndez Muñoz, D. (2021) 'Evaluation Metrics in Model Transformation: A Systematic Review', *Journal of Software Engineering Research and Development*, 9(1), pp. 1–24.

16. Zhou, M., Peng, H., Chen, Q. et al. (2023) 'Domain-specific adaptation of large language models: A survey', *ACM Computing Surveys*.[online] Διαθέσιμο στο: https://arxiv.org/abs/2307.10174 [Πρόσβαση 22 Απρ. 2025].

17. Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L. και Chen, W. (2021) *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv preprint arXiv:2106.09685. [online] Διαθέσιμο στο: https://arxiv.org/abs/2106.09685 [Πρόσβαση 10 Ιουν. 2025].

18. Dettmers, T., Pagnoni, A., Holtzman, A. και Zettlemoyer, L. (2022) *Efficient Fine-Tuning of Language Models: Parameter-Efficient Fine-Tuning Methods*. arXiv preprint arXiv:2203.15556. [online] Διαθέσιμο στο: https://arxiv.org/abs/2203.15556 [Πρόσβαση 10 Ιουν. 2025].