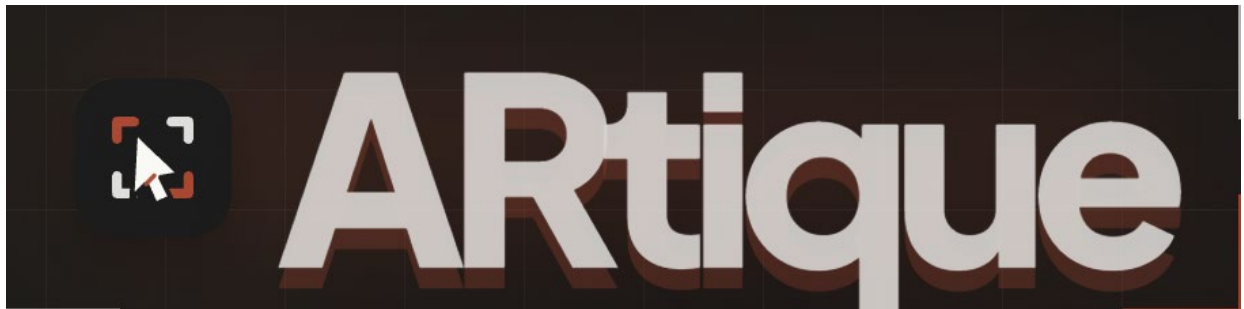


ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
«Κριτικός σχεδιασμού εφαρμογής με χρήσης
επαυξημένης πραγματικότητας (ARtique)»



Του φοιτητή
Χαματζόγλου Δημήτριος
Αρ. Μητρώου: iee2020258

Επιβλέπων
Ονοματεπώνυμο Ευκλείδης
Κεραμόπουλος
Καθηγητής

Ημερομηνία 14/06/2026

Τίτλος Δ.Ε. **Κριτικός σχεδιασμού εφαρμογής με χρήσης επαυξημένης πραγματικότητας**

Κωδικός Δ.Ε. **26259**

Ονοματεπώνυμο φοιτητή **Χαμαμτζόγλου Δημήτριος**

Ονοματεπώνυμο εισηγητή **Ευκλείδης Κεραμόπουλος**

Ημερομηνία ανάληψης Δ.Ε. 13/03/2026

Ημερομηνία περάτωσης Δ.Ε. 6/9/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Δημήτριου Χαμαμτζόγλου που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Στη μνήμη του πατέρα μου.

Στη μητέρα μου, την αδερφή μου και τη Δάφνη, που ήταν πάντα εκεί.

Και σε κάθε παιδί που μεγάλωσε με μια απουσία και κράτησε ζωντανό αυτό που ονειρεύτηκε.

Πρόλογος

Η επιλογή της παρούσας διπλωματικής εργασίας προέκυψε από την επιθυμία μου να ασχοληθώ με τεχνολογίες πέρα από την κλασική ανάπτυξη web εφαρμογών. Τα τελευταία τρία χρόνια εργάζομαι ως full stack developer. Η εργασία αυτή ήταν μια ευκαιρία να μελετήσω την τεχνητή νοημοσύνη και την επαυξημένη πραγματικότητα στο web. Στο ARtique μελέτησα τη χρήση του YOLO για τον εντοπισμό στοιχείων διεπαφής. Εξέτασα επίσης πώς τα στοιχεία αυτά μπορούν να χρησιμοποιηθούν ως σημεία αγκύρωσης για τις παρατηρήσεις της αξιολόγησης. Η προβολή των ευρημάτων γίνεται στο περιβάλλον WebAR.

Κατά την υλοποίηση ασχολήθηκα με την αξιολόγηση UI/UX και με την ενσωμάτωση μοντέλων τεχνητής νοημοσύνης σε μια web εφαρμογή. Η διαδικασία ανέδειξε και τους περιορισμούς που εμφανίζονται όταν μια ιδέα μετατρέπεται σε λειτουργικό πρωτότυπο.

Περίληψη

Η παρούσα διπλωματική εργασία αφορά τον σχεδιασμό και την ανάπτυξη του Web AR Design Critic, μιας web εφαρμογής για την αξιολόγηση διεπαφών χρήστη από εικόνα που λαμβάνεται με την κάμερα μιας κινητής συσκευής. Η εφαρμογή αναλύει την ορατή διεπαφή και επιστρέφει παρατηρήσεις για ζητήματα UI/UX.

Το frontend υλοποιήθηκε με Angular. Το backend βασίζεται σε Node.js και Express, ενώ τα αποτελέσματα των αξιολογήσεων αποθηκεύονται σε MySQL. Το Gemini αξιολογεί την εικόνα της διεπαφής. Το YOLO εντοπίζει στοιχεία της εικόνας και παρέχει τις θέσεις τους για την αγκύρωση των ευρημάτων. Η απόκριση της ανάλυσης επιστρέφεται σε μορφή JSON. Για κάθε εύρημα περιγράφεται το ζήτημα και προτείνεται σχετική βελτίωση. Η απόκριση περιλαμβάνει επίσης επίπεδο σοβαρότητας και βαθμό εμπιστοσύνης. Όταν ένα εύρημα αντιστοιχίζεται σε ανιχνευμένο στοιχείο, η εφαρμογή το προβάλλει με πλαίσιο στην εικόνα. Το πρωτότυπο παρουσιάζει τα αποτελέσματα σε γραπτή αναφορά και στη ζωντανή προβολή της διεπαφής. Η αξιολόγηση περιορίζεται σε ζητήματα που είναι ορατά στην εικόνα. Τα ευρήματα έχουν συμβουλευτικό χαρακτήρα και δεν αποτελούν πλήρη πιστοποίηση προσβασιμότητας ή ποιότητας.

«Application Design Critic Using Augmented Reality (ARtique)»

«Dimitrios Chamamtzoglou»

Abstract

This thesis presents ARtique, a web application that reviews a website as it appears on a real screen. A mobile device camera captures the screen, and the application produces observations about the visible user interface. The review is based on what can be seen in the image. This includes readability, contrast, layout, spacing, and the apparent usability of interactive elements. The findings are shown both in a written report and as markers over the parts of the screen where the issues appear.

For each issue, the report includes a severity level and a confidence estimate. If the issue is connected to a detected interface element, the result also includes its location. The report may also suggest a change to the interface. This project was made using Angular for the frontend, Node.js and Express for the backend, and MySQL to store and manage the results, along with an AI model that analyses images and finds any usability problems with the website, and YOLO models to detect UI elements such as buttons and menus so that the website can be bounded on screen to highlight any issues that were found. This application can give the user a general idea of the usability problems that may exist on a website. It is not perfect and will make its share of mistakes, but it is very effective at its task. The main feature of this application is being able to point your camera to a website and get a report on the usability of the website. Although, I would like to work on improving the AI model and fix any and all bugs that still linger in the application, the project is a prototype for a real-life use-case and is built using web AR, computer vision, machine learning, and AI to create a fully working application.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς τον επιβλέποντα της διπλωματικής μου εργασίας, κ. Ευκλείδη Κεραμόπουλο, για την καθοδήγηση, την επιστημονική υποστήριξη και τις χρήσιμες παρατηρήσεις του καθ' όλη τη διάρκεια της εκπόνησής της. Η καθοδήγησή του επηρέασε την πορεία της εργασίας και με βοήθησε να την ολοκληρώσω.

Επιπλέον, θα ήθελα να ευχαριστήσω το Διεθνές Πανεπιστήμιο Ελλάδας και το Τμήμα μου για τις γνώσεις, τα εφόδια και τις εμπειρίες που μου προσέφεραν κατά τη διάρκεια των σπουδών μου, τα οποία συνέβαλαν ουσιαστικά στην υλοποίηση της παρούσας διπλωματικής εργασίας.

Ευχαριστώ την οικογένειά μου και τους κοντινούς μου ανθρώπους για τη στήριξη που μου έδειξαν κατά την εκπόνηση της εργασίας. Η παρουσία και η συμβολή τους αποτέλεσαν σημαντικό παράγοντα για την ολοκλήρωση της εργασίας.

Περιεχόμενα

Πρόλογος	i
Περίληψη	ii
Abstract	iii
Ευχαριστίες	iv
Περιεχόμενα	v
Κατάλογος Σχημάτων	vii
Κατάλογος Πινάκων	viii
Συνοτομογραφίες	ix
Κεφάλαιο 1: Εισαγωγή	1
1.1 Εισαγωγή	1
1.2 Προσδιορισμός του Προβλήματος.....	1
1.3 Κίνητρο και Στόχοι της Διπλωματικής Εργασίας.....	2
1.4 Δομή της Διπλωματικής Εργασίας	2
1.5 Επίλογος	3
Κεφάλαιο 2: Θεωρητικό Υπόβαθρο και Σχετικές Προσεγγίσεις στην AI-Assisted Αξιολόγηση Διεπαφών	5
2.1 Εισαγωγή	5
2.2 Ανάλυση Υφιστάμενων Προσεγγίσεων και Εφαρμογών.....	5
2.2.1 UICrit	5
2.2.2 Visual Prompting with Iterative Refinement for Design Critique Generation	6
2.3 Δυνατά και Αδύναμα Σημεία Ανταγωνιστικών Λύσεων	7
2.4 Κενά της Αγοράς και Καινοτομία της Προτεινόμενης Εφαρμογής.....	8
2.4.1 Κενά της Αγοράς.....	8
2.4.2 Καινοτομία της Προτεινόμενης Εφαρμογής	9
2.5 Επίλογος	9
Κεφάλαιο 3: Μεθοδολογία και Σχεδιαστικές Αποφάσεις	11
3.1 Εισαγωγή	11
3.2 Επιλογή Γλωσσών Προγραμματισμού.....	11
3.2.1 TypeScript.....	12
3.2.2 JavaScript και Nodejs.....	13
3.2.3 Python.....	14
3.2.4 JSON και SQL για ανταλλαγή και αποθήκευση δεδομένων	15
3.3 Κριτήρια Επιλογής Τεχνολογιών.....	16
3.4 Συνολική Αρχιτεκτονική Συστήματος	18
3.5 Μηχανική Μάθηση και Υπολογιστική Όραση	20
3.6 Backend και API Υποδομή.....	21

3.7 Μεθοδολογία AI UI/UX Αξιολόγησης	23
3.8 Αποθήκευση Δεδομένων	24
3.9 Βασικές Σχεδιαστικές Αποφάσεις και Περιορισμοί	26
3.10 Συνοπτική αποτίμηση της αρχιτεκτονικής.....	26
3.11 Επίλογος	27
Κεφάλαιο 4: Τεχνική Υλοποίηση του ARtique.....	29
4.1 Εισαγωγή	29
4.2 Ανάπτυξη Back-End	29
4.2.1 Οργάνωση Δομής Back-End	29
4.2.2 API Endpoints και Ροή Ανάλυσης	30
4.2.3 Επεξεργασία Εικόνας και Επικοινωνία με Gemini.....	31
4.2.4 Κανονικοποίηση, Anchoring και Αποθήκευση	34
4.3 Ανάπτυξη Front-End.....	36
4.3.1 Αρχιτεκτονική και Οντότητες Front-End	37
4.3.2 Ροή Σάρωσης και Διαχείριση Κάμερας.....	37
4.3.3 Live WebAR Pipeline και Ανίχνευση Οθόνης	38
4.3.4 Ανίχνευση Στοιχείων UI με YOLO.....	40
4.3.5 Οθόνες Διεπαφής Χρήστη και Περιπτώσεις Χρήσης.....	42
4.4 Έλεγχοι συνέπειας και περιορισμοί υλοποίησης	55
4.5 Επίλογος	56
Κεφάλαιο 5: Συμπεράσματα και Μελλοντική Εργασία	57
5.1 Εισαγωγή	57
5.2 Πλαίσιο ερμηνείας των αποτελεσμάτων.....	58
5.3 Συμπεράσματα	59
5.4 Μελλοντική εργασία και βελτιστοποίηση	60
5.5 Επίλογος	62
Βιβλιογραφία	63

Κατάλογος Σχημάτων

Σχήμα 3.1: Επίσημο λογότυπο της TypeScript.	12
Σχήμα 3.2: Επίσημο λογότυπο του Node.js.	14
Σχήμα 3.3: Επίσημο λογότυπο του Express.js.	14
Σχήμα 3.4: Επίσημο λογότυπο της Python.....	15
Σχήμα 3.5: Επίσημο λογότυπο του Angular.	16
Σχήμα 3.6: Επίσημο λογότυπο του OpenCV, του οποίου η έκδοση OpenCV.js χρησιμοποιείται για screen detection.	17
Σχήμα 3.7: Επίσημο λογότυπο του Ultralytics YOLOv8 που χρησιμοποιήθηκε για την εκπαίδευση του UI-element detector.....	17
Σχήμα 3.8: Επίσημο λογότυπο του ONNX Runtime, μέσω του οποίου εκτελείται το μοντέλο στον browser.	18
Σχήμα 3.9: Επίσημο οπτικό σήμα του Gemini API που χρησιμοποιείται για τη δομημένη UI/UX αξιολόγηση..	18
Σχήμα 3.10: Συνολική αρχιτεκτονική και ροή δεδομένων της εφαρμογής ARtique.....	19
Σχήμα 4.1: Δήλωση των endpoints των audits και χρήση του middleware μεταφόρτωσης εικόνας.....	30
Σχήμα 4.2: Παραλαβή εικόνας και στοιχείων uiElementsJson από το request.	31
Σχήμα 4.3: Δημιουργία του multipart request από το Angular front-end.	31
Σχήμα 4.4: Προεπεξεργασία και συμπίεση εικόνας πριν την αποστολή στο AI μοντέλο.....	32
Σχήμα 4.5: Επιλογή λειτουργίας AI και κλήση της υπηρεσίας Gemini.	33
Σχήμα 4.6: Απόσπασμα του anchor-first prompt που δίνει στο Gemini τα detected UI elements.	34
Σχήμα 4.7: Διόρθωση και επαλήθευση anchoring των AI issues.....	35
Σχήμα 4.8: Κανονικοποίηση του UX checklist με τις έντεκα σταθερές κατηγορίες.....	35
Σχήμα 4.9: Δημιουργία των βασικών πινάκων MySQL για audits, audit_issues και audit_feedback.	36
Σχήμα 4.10: Κατάσταση Ready to scan με κλειδωμένο screen quad και ορατά στοιχεία διεπαφής σε πραγματική κινητή συσκευή.	38
Σχήμα 4.11: Ανίχνευση οθόνης στο live camera feed.....	39
Σχήμα 4.12: Μηχανισμός σταθεροποίησης του screen quad με deadband, freeze και relock.....	39
Σχήμα 4.13: Έλεγχος κίνησης κάμερας που επιτρέπει το πάγωμα του quad όταν η συσκευή είναι ακίνητη.	40
Σχήμα 4.14: Εκτέλεση YOLO στο browser μέσω ONNX Runtime Web.....	41
Σχήμα 4.15: Tracking και εξομάλυνση των YOLO bounding boxes.....	41
Σχήμα 4.16: Παραγωγή camera-augmented dataset για καλύτερη προσαρμογή σε λήψεις οθόνης από κινητό...	42
Σχήμα 4.17: Αρχική σελίδα του ARtique σε επιτραπέζια προβολή.	44
Σχήμα 4.18: Αρχική σελίδα του ARtique σε πραγματική κινητή συσκευή.....	45
Σχήμα 4.19: Επιλογή τρόπου σάρωσης σε επιτραπέζια προβολή.	46
Σχήμα 4.20: Επιλογή τρόπου σάρωσης σε πραγματική κινητή συσκευή.....	47
Σχήμα 4.21: Επιλογή δοκιμαστικών ή πραγματικών σχολίων AI σε επιτραπέζια προβολή.	48
Σχήμα 4.22: Επιλογή λειτουργίας σχολίων σε πραγματική κινητή συσκευή.....	49
Σχήμα 4.23: Πύλη εισαγωγής κωδικού για την πραγματική ανάλυση, χωρίς ανοικτό πληκτρολόγιο.	50
Σχήμα 4.24: Οδηγία αλλαγής σε landscape προσανατολισμό πριν από τη ζωντανή σάρωση.	51
Σχήμα 4.25: Κατάσταση ανάλυσης UI/UX ενώ το live camera view παραμένει ενεργό.....	52
Σχήμα 4.26: Προβολή του UX Report με τις καταστάσεις Good και Issue.	52
Σχήμα 4.27: Λεπτομέρεια επιλεγμένου issue με κατηγορία, περιγραφή και πρόταση βελτίωσης.....	53
Σχήμα 4.28: Πλήρης αναφορά πραγματικού audit με αγκυρωμένα ευρήματα και σύνοψη.....	54
Σχήμα 4.29: Φορτωμένο ιστορικό αποθηκευμένων audits με προεπισκοπήσεις και κατανομή ζητημάτων.....	55

Κατάλογος Πινάκων

Πίνακας 3.1: Βασικά API endpoints της εφαρμογής	22
Πίνακας 3.2: Βασικοί πίνακες της βάσης δεδομένων	25

Συντομογραφίες

Δ.Ε. — Διπλωματική Εργασία

ΔΙΠΙΑΕ — Διεθνές Πανεπιστήμιο της Ελλάδος

ACM — Association for Computing Machinery — Διεθνής επιστημονικός οργανισμός πληροφορικής

AI — Artificial Intelligence — Τεχνητή Νοημοσύνη

API — Application Programming Interface — Διεπαφή Προγραμματισμού Εφαρμογών

AR — Augmented Reality — Επαυξημένη Πραγματικότητα

ARIA — Accessible Rich Internet Applications — Προδιαγραφή προσβασιμότητας για διαδικτυακές εφαρμογές

CSS — Cascading Style Sheets — Διαδοχικά Φύλλα Στυλ

DOM — Document Object Model — Μοντέλο Αντικειμένου Εγγράφου

EMA — Exponential Moving Average — Εκθετικός Κινητός Μέσος

HTML — HyperText Markup Language — Γλώσσα Σήμανσης Υπερκειμένου

JPEG — Joint Photographic Experts Group — Μορφή συμπίεσης ψηφιακών εικόνων

JSON — JavaScript Object Notation — Μορφή ανταλλαγής δομημένων δεδομένων

MySQL — My Structured Query Language — Σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων

ONNX — Open Neural Network Exchange — Ανοιχτή μορφή ανταλλαγής μοντέλων νευρωνικών δικτύων

OpenCV — Open Source Computer Vision Library — Βιβλιοθήκη υπολογιστικής όρασης ανοικτού κώδικα

SDK — Software Development Kit — Πακέτο ανάπτυξης λογισμικού

SQL — Structured Query Language — Δομημένη Γλώσσα Ερωτημάτων

UI — User Interface — Διεπαφή Χρήστη

UI/UX — User Interface / User Experience — Διεπαφή και Εμπειρία Χρήστη

UIST — ACM Symposium on User Interface Software and Technology — Συνέδριο για λογισμικό και τεχνολογία διεπαφών

UX — User Experience — Εμπειρία Χρήστη

W3C — World Wide Web Consortium — Κοινοπραξία του Παγκόσμιου Ιστού

WAVE — Web Accessibility Evaluation Tool — Εργαλείο αξιολόγησης προσβασιμότητας ιστοσελίδων

WCAG — Web Content Accessibility Guidelines — Οδηγίες Προσβασιμότητας Περιεχομένου Ιστού

WebAR — Web-based Augmented Reality — Επαυξημένη Πραγματικότητα μέσω φυλλομετρητή

WebXR — Web Extended Reality — Πρότυπο εκτεταμένης πραγματικότητας για τον ιστό

YOLO — You Only Look Once — Οικογένεια μοντέλων ανάγνωσης αντικειμένων

Κεφάλαιο 1: Εισαγωγή

1.1 Εισαγωγή

Σήμερα όλοι μας περνάμε πολύ χρόνο σε ιστοσελίδες και εφαρμογές για να ψάξουμε προϊόντα ή για να μάθουμε κάτι καινούργιο. Η ποιότητα της διεπαφής χρήστη δεν επηρεάζει μόνο την εμφάνιση μιας εφαρμογής, αλλά και το πώς οι μέσοι άνθρωποι την κατανοούν και τη χρησιμοποιούν. Οι σχεδιαστές, οι προγραμματιστές και οι ειδικοί σε θέματα ευχρηστίας προσπαθούν συνεχώς να βρουν τρόπους να βελτιώσουν τις διεπαφές και την εμπειρία του χρήστη, όπως το πόσο εύκολο είναι να διαβάσεις κάτι, τα χρώματα και η διάταξη των στοιχείων. Αυτή η δουλειά παίρνει πάντα πολύ χρόνο και απαιτεί μεγάλη γνώση επάνω στο πεδίο του σχεδιασμού. [1], [2]

Τώρα που η τεχνητή νοημοσύνη και η μηχανική μάθηση γίνονται όλο και πιο έξυπνες, υπάρχουν νέοι τρόποι για να να βρούμε λύσεις σε προβλήματα που παλιά θα θεωρούνταν αδύνατο να αυτοματοποιηθούν. Σε αυτό το πρόγραμμα, προσπαθούμε να ενώσουμε τις τεχνολογίες του διαδικτύου, την τεχνητή νοημοσύνη και την επαυξημένη πραγματικότητα. Θέλουμε ουσιαστικά η τεχνητή νοημοσύνη και η επαυξημένη πραγματικότητα να βοηθήσουν στην αξιολόγηση της διεπαφής του χρήστη. [3]

Φτιάχτηκε ένα πρόγραμμα που το ονομάζω ARtique, μέσα στο οποίο ο χρήστης μπορεί να στοχεύει μια ιστοσελίδα χρησιμοποιώντας την κάμερα της συσκευής του και να βλέπει πληροφορίες για τα πιθανά προβλήματα εμπειρίας και διεπαφής του χρήστη, εμφανίζοντας μια αναφορά και με σημάδευμένα τα προβλήματα πάνω στην εικόνα της κάμερας

1.2 Προσδιορισμός του Προβλήματος

Κατά την ανάπτυξη μιας web εφαρμογής, ο δημιουργός της δεν εντοπίζει πάντοτε αμέσως τα προβλήματα UI/UX. Η μακρόχρονη ενασχόληση με την ίδια διεπαφή κάνει τις ενέργειές της γνώριμες στον δημιουργό. Ο νέος χρήστης συναντά τη διεπαφή χωρίς να γνωρίζει εκ των προτέρων τον τρόπο λειτουργίας της. Κατά την ανάπτυξη μπορεί να μη γίνει αντιληπτό ότι το κείμενο δεν ξεχωρίζει αρκετά από το φόντο. Το ίδιο μπορεί να συμβεί με ένα κουμπί που είναι δύσχρηστο σε κινητή συσκευή. Μια αξιολόγηση από τεχνητή νοημοσύνη ή ειδικό σχεδίασης μπορεί να εντοπίσει σωστά ένα ζήτημα. Η αναφορά χρειάζεται να δείχνει και το στοιχείο στο οποίο αναφέρεται. Όταν περιορίζεται σε μια γενική διατύπωση, όπως «πρόβλημα αντίθεσης» ή «λανθασμένη ευθυγράμμιση», ο χρήστης χρειάζεται να εντοπίσει μόνος του το σχετικό σημείο στη σελίδα. Σε μια σύνθετη διεπαφή, ο χρήστης ίσως δυσκολευτεί να αντιστοιχίσει την παρατήρηση με το σωστό στοιχείο.

Γι' αυτό το ARtique διατηρεί τη γραπτή αναφορά, αλλά τη συνδέει επίσης με την εικόνα που εξετάζει ζωντανά υιοθετώντας έτσι τον AR χαρακτήρα της εφαρμογής. Μετά τη σάρωση, το σύστημα συνδέει τα ευρήματα με τα στοιχεία που έχει εντοπίσει στην οθόνη. Όταν υπάρχει αντιστοίχιση, εμφανίζεται πλαίσιο στο αντίστοιχο σημείο της οθόνης. Ο χρήστης βλέπει την παρατήρηση στο στοιχείο στο οποίο αναφέρεται. Η ανάλυση μιας εικόνας έχει συγκεκριμένα όρια. Η εικόνα δεν αρκεί για πλήρη έλεγχο της δομής του DOM και των ARIA attributes. Ο έλεγχος με πληκτρολόγιο ή screen reader δεν προκύπτει

από την εικόνα. Η εφαρμογή καταγράφει μόνο ζητήματα που διακρίνονται στην εικόνα. Τα ευρήματά της έχουν συμβουλευτικό χαρακτήρα και δεν ισοδυναμούν με πιστοποίηση προσβασιμότητας ή ποιότητας. [2], [4]

1.3 Κίνητρο και Στόχοι της Διπλωματικής Εργασίας

Η εργασία ξεκίνησε από το ερώτημα πώς μπορεί μια web εφαρμογή να συνδυάσει AI-based ανάλυση, μηχανική μάθηση και Web AR. Το ενδιαφέρον στράφηκε στη σύνδεση της αρχιτεκτονικής frontend-backend με την επεξεργασία εικόνας, την ανίχνευση UI elements και την προβολή των αποτελεσμάτων πάνω στη ζωντανή εικόνα της κάμερας.

Ο πρώτος στόχος ήταν να μπορεί η εφαρμογή να καταγράφει μια ορατή web διεπαφή μέσα από την κάμερα μιας κινητής συσκευής και να παράγει δομημένο UI/UX audit. Η εικόνα αποστέλλεται στο Gemini, όπου αναλύονται τα οπτικά χαρακτηριστικά της διεπαφής και επιστρέφεται αποτέλεσμα σε JSON. Για κάθε issue καταγράφονται η κατηγορία, η σοβαρότητα, ο βαθμός εμπιστοσύνης και μια προτεινόμενη βελτίωση.

Ο δεύτερος στόχος αφορούσε τη χρήση μοντέλου YOLO για την αναγνώριση ορατών στοιχείων της διεπαφής. Τα bounding boxes που παράγει το μοντέλο χρησιμοποιούνται ως spatial anchors για τα αποτελέσματα του audit. Όταν ένα issue αντιστοιχίζεται με ασφάλεια σε κάποιο από αυτά τα στοιχεία, μπορεί να προβληθεί στην αντίστοιχη θέση του camera view. Με αυτόν τον τρόπο, ο χρήστης δεν λαμβάνει μόνο μια περιγραφή του προβλήματος, αλλά μπορεί να δει και την περιοχή στην οποία αναφέρεται.

Ο τρίτος στόχος ήταν η υλοποίηση ολόκληρης της διαδρομής ενός audit σε λειτουργικό πρωτότυπο. Η διαδρομή ξεκινά από τη λήψη και τη ζωντανή ανίχνευση στο Angular frontend, συνεχίζεται με την επεξεργασία της εικόνας και τον έλεγχο των anchors στο Node.js/Express backend και ολοκληρώνεται με την αποθήκευση του αποτελέσματος στη MySQL. Το πρωτότυπο παρέχει έτσι μια συγκεκριμένη βάση για τις δοκιμές και την αξιολόγηση της προσέγγισης. Χρειάζεται έλεγχος ότι οι πηγές [2] και [4] τεκμηριώνουν πράγματι τους συγκεκριμένους περιορισμούς της αξιολόγησης διεπαφών αποκλειστικά μέσω εικόνας.

1.4 Δομή της Διπλωματικής Εργασίας

Το Κεφάλαιο 1 εισάγει το αντικείμενο της εργασίας και το πρόβλημα που εξετάζεται. Παρουσιάζει το κίνητρο για την ανάπτυξη της εφαρμογής, μαζί με τους στόχους της διπλωματικής. Το Κεφάλαιο 2 εστιάζει στη θεωρητική θεμελίωση και τη βιβλιογραφική ανασκόπηση των βασικών πυλώνων της μελέτης. Συγκεκριμένα, αναλύονται οι αρχές αξιολόγησης διεπαφών, εμπειρίας χρήστη (UI/UX) και προσβασιμότητας, ενώ παράλληλα εξετάζεται το τεχνολογικό υπόβαθρο που πλαισιώνει την εργασία, με έμφαση στην τεχνητή νοημοσύνη, τη μηχανική μάθηση, τα μοντέλα ανίχνευσης αντικειμένων YOLO και τις τεχνολογίες επαυξημένης πραγματικότητας στον ιστό (WebAR).. Παράλληλα, παρουσιάζονται οι περιορισμοί της ανάλυσης διεπαφών αποκλειστικά μέσω εικόνας.

Στο Κεφάλαιο 3 περιγράφεται ο τρόπος με τον οποίο αναπτύχθηκε η εφαρμογή. Παρουσιάζονται οι σχεδιαστικές επιλογές που έγιναν κατά την υλοποίησή της. Εξετάζεται η αρχιτεκτονική του συστήματος και αιτιολογείται η επιλογή των τεχνολογιών. Περιγράφεται η ροή των δεδομένων από την κάμερα έως την ανάλυση από το AI, καθώς και η μορφή με την οποία επιστρέφονται τα αποτελέσματα στην εφαρμογή.

Στο Κεφάλαιο 4 περιγράφεται αναλυτικά η τεχνική υλοποίηση του Web AR Design Critic. Περιγράφεται η υλοποίηση του frontend με Angular και του backend με Node.js/Express. Εξηγείται επίσης η χρήση της MySQL για την αποθήκευση των αποτελεσμάτων. Στη συνέχεια περιγράφεται η σύνδεση με το Gemini, καθώς και ο τρόπος με τον οποίο το YOLO εντοπίζει στοιχεία της διεπαφής που χρησιμοποιούνται στις οπτικές επικαλύψεις.

Στο Κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα της εργασίας με βάση την υλοποίηση του συστήματος. Περιγράφονται οι βασικοί περιορισμοί που εμφανίστηκαν κατά τη χρήση του πρωτοτύπου. Εξετάζονται επίσης τα επόμενα βήματα της εφαρμογής. Η ανάγκη για ακριβέστερα μοντέλα ανίχνευσης είναι μία από τις κατευθύνσεις που προτείνονται. Η αξιολόγηση των αποτελεσμάτων χρειάζεται να επεκταθεί σε περισσότερες περιπτώσεις χρήσης. Οι Web AR λειτουργίες μπορούν να εξελιχθούν περαιτέρω.

1.5 Επίλογος

Η εργασία εξετάζει πώς οι παρατηρήσεις για μια διεπαφή μπορούν να συνδεθούν με τα ορατά στοιχεία στα οποία αναφέρονται. Έτσι, ο χρήστης μπορεί να βλέπει σε ποιο σημείο της διεπαφής αντιστοιχεί κάθε παρατήρηση.

Με βάση αυτούς τους στόχους, το επόμενο κεφάλαιο εξετάζει τις υπάρχουσες μεθόδους αξιολόγησης διεπαφών και τα όρια της ανάλυσης που βασίζεται αποκλειστικά σε εικόνα.

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο και Σχετικές Προσεγγίσεις στην AI-Assisted Αξιολόγηση Διεπαφών

2.1 Εισαγωγή

Το κεφάλαιο παρουσιάζει τρόπους αξιολόγησης διεπαφών που σχετίζονται με την εργασία, όπως εργαλεία προσβασιμότητας, μέθοδοι οπτικής ανάλυσης και λύσεις που χρησιμοποιούν μοντέλα τεχνητής νοημοσύνης. Η σύγκριση γίνεται με βάση τα δεδομένα που εξετάζει κάθε προσέγγιση και τον τρόπο με τον οποίο εμφανίζει τα ευρήματα στον χρήστη.

Η ανασκόπηση δεν αντιμετωπίζει την ανάλυση εικόνας ως υποκατάστατο του ελέγχου κώδικα ή της ανθρώπινης κρίσης. Στόχος είναι να οριοθετηθεί το πεδίο στο οποίο το ARtique επιχειρεί να συνδυάσει οπτική αξιολόγηση, χωρική αγκύρωση και αναφορά αποτελεσμάτων.

2.2 Ανάλυση Υφιστάμενων Προσεγγίσεων και Εφαρμογών

Για να κατανοήσουμε τον τρόπο με τον οποίο αντιμετωπίζεται σήμερα η αυτοματοποιημένη αξιολόγηση διεπαφών χρήστη, χρειάζεται πρώτα να αναλύσουμε τις ήδη υπάρχουσες προσεγγίσεις που έχουν υλοποιηθεί. Συγκεκριμένα, η αξιολόγηση του UI/UX, δηλαδή του User Interface και του User Experience, δεν περιορίζεται μόνο στον εντοπισμό τεχνικών σφαλμάτων [2], αλλά αφορά και ζητήματα όπως η οπτική ιεραρχία, η αναγνωσιμότητα, η αντίθεση, η ευχρηστία, η προσβασιμότητα και η σαφήνεια των στοιχείων αλληλεπίδρασης. Οι λύσεις που εξετάζονται βασίζονται συνήθως σε ανάλυση του κώδικα και του DOM ή σε στατικά screenshots. Οι προσεγγίσεις αυτές καλύπτουν διαφορετικά μέρη της αξιολόγησης. Η προτεινόμενη εφαρμογή καταγράφει τη διεπαφή με την κάμερα και προβάλλει τα ευρήματα πάνω στην εικόνα.

Στο πλαίσιο αυτό εξετάζονται:

- Στην κατηγορία αυτή περιλαμβάνονται το UICrit και το Visual Prompting with Iterative Refinement for Design Critique Generation. Οι προσεγγίσεις αυτές παράγουν UI critiques από screenshots και χρησιμοποιούν bounding boxes για τη σύνδεση των σχολίων με περιοχές της εικόνας. [1], [3]
- εργαλεία προσβασιμότητας όπως τα WAVE και axe DevTools, τα οποία εντοπίζουν προβλήματα accessibility κυρίως μέσω browser-based ή DOM-based ανάλυσης, [5], [6]
- εργαλεία οπτικής ανάλυσης όπως το Attention Insight, που χρησιμοποιούν AI για την πρόβλεψη περιοχών προσοχής μέσω heatmaps, [7]

Η σύγκριση με τις παραπάνω λύσεις εστιάζει στη χρήση κάμερας στην προτεινόμενη εφαρμογή. Η εφαρμογή συνδέει τα ευρήματα της αξιολόγησης με οπτικές επισημάνσεις στην ορατή διεπαφή.

2.2.1 UICrit

Το «UICrit: Enhancing Automated Design Evaluation with a UI Critique Dataset», το οποίο παρουσιάστηκε για πρώτη φορά στο ACM UIST 2024, αποτελεί μία από τις ακαδημαϊκές εργασίες που σχετίζονται περισσότερο με την παρούσα διπλωματική εργασία, καθώς εξετάζει την αυτοματοποιημένη

αξιολόγηση διεπαφών χρήστη με τη χρήση μεγάλων γλωσσικών μοντέλων και οπτικής πληροφορίας. Η προσέγγιση βασίζεται σε ένα σύνολο δεδομένων που περιλαμβάνει στιγμιότυπα οθόνης διεπαφών χρήστη, σχόλια αξιολόγησης σχεδιασμού διατυπωμένα σε φυσική γλώσσα, πλαίσια οριοθέτησης για τις περιοχές στις οποίες αναφέρονται τα σχόλια, καθώς και βαθμολογίες σχετικά με την ποιότητα του σχεδιασμού. [1]

Οι συγγραφείς του UICrit επισημαίνουν ότι ένα από τα βασικά προβλήματα των μεγάλων γλωσσικών μοντέλων είναι ότι έχουν σχεδιαστεί ώστε να γενικεύουν σε πολλά διαφορετικά είδη εργασιών. Ως αποτέλεσμα, η παραγωγή ουσιαστικών και χρήσιμων αξιολογήσεων UI/UX παραμένει δύσκολη. Μια τέτοια αξιολόγηση δεν περιορίζεται απλώς στην περιγραφή του περιεχομένου μιας εικόνας, αλλά απαιτεί την κατανόηση βασικών αρχών σχεδιασμού, όπως η προσβασιμότητα, η οπτική ιεραρχία, η αντίθεση, η ευχρηστία και η οπτική σαφήνεια [1]

Στην αρχική δημοσίευση, το UICrit περιλάμβανε 3.059 design critiques για 983 mobile UI screens, τα οποία συλλέχθηκαν από έμπειρους designers. Στη δημόσια έκδοση του dataset στο GitHub αναφέρεται μεταγενέστερη, διευρυμένη έκδοση με 11.344 critiques για περίπου 1.000 mobile UIs. Κάθε αξιολόγηση συνδέεται με συγκεκριμένη περιοχή της διεπαφής μέσω bounding box, επιτρέποντας στο μοντέλο να εντοπίζει και πού ακριβώς εμφανίζεται το πρόβλημα. [1]

Οι συγγραφείς της εργασίας χρησιμοποιούν το dataset για να βελτιωθεί η απόδοση multimodal LLMs σε εργασίες UI feedback generation δοκιμάζοντας few-shot prompting και visual prompting τεχνικές, δηλαδή παρέχουν στο μοντέλο παραδείγματα παρόμοιων διεπαφών και οπτικές ενδείξεις ώστε να εκτιμήσει καλύτερα τα προβλήματα και τις αντίστοιχες περιοχές και σύμφωνα με τα αποτελέσματα η χρήση του UICrit οδηγεί σε σημαντική βελτίωση της ποιότητας των AI-generated critiques σε σχέση με zero-shot prompting. [1]

Ο UICrit είναι κάτι τελείως διαφορετικό από αυτή την εφαρμογή. Λειτουργεί κυρίως ως εργαλείο για έρευνα και πειράματα, όπου παράγει κριτικές από στατικές εικόνες οθονών. Δεν έχει καμία σχέση με τη χρήση κάμερας σε κινητά. Δεν εξετάζει οθόνες σε πραγματικά περιβάλλοντα και δεν δείχνει τα αποτελέσματα σαν να είναι πάνω στην πραγματική οθόνη. Αντίθετα, η δουλειά μας φέρνει την ιδέα της κριτικής βασισμένης σε εικόνες οθονών σε μια πιο πρακτική χρήση. Ο χρήστης στρέφει την κάμερα του κινητού του προς μια οθόνη. Η εφαρμογή καταγράφει την ορατή διεπαφή και δίνει κριτική με εικόνες που τοποθετούνται πάνω στα αποτελέσματα.

2.2.2 Visual Prompting with Iterative Refinement for Design Critique Generation

Η εργασία Visual Prompting with Iterative Refinement for Design Critique Generation επεκτείνει την προσέγγιση του UICrit και εστιάζει περισσότερο στη βελτίωση της ίδιας της διαδικασίας παραγωγής UI critiques και bounding boxes. Η εργασία προτείνει ένα pipeline βασισμένο σε multimodal LLMs, το οποίο δέχεται ως είσοδο ένα UI screenshot και ένα σύνολο design guidelines, και παράγει ως έξοδο

λίστα από design comments μαζί με bounding boxes που αντιστοιχούν στις περιοχές της εικόνας όπου εντοπίζεται κάθε πρόβλημα. [3]

Η βασική συνεισφορά της εργασίας είναι η χρήση iterative refinement. Αντί το μοντέλο να παράγει μία τελική απάντηση σε ένα μόνο βήμα, η διαδικασία χωρίζεται σε επιμέρους στάδια. Αρχικά παράγονται design comments, στη συνέχεια φιλτράρονται τα μη έγκυρα ή λιγότερο χρήσιμα σχόλια, έπειτα παράγονται bounding boxes και τέλος τα σχόλια και οι περιοχές ελέγχονται και βελτιώνονται επαναληπτικά. Η αρχιτεκτονική που περιγράφεται στην εργασία χρησιμοποιεί πολλαπλά LLM components για παραγωγή, φιλτράρισμα, validation και refinement. [3]

Οι ερευνητές υπογραμμίζουν ότι τα multimodal LLMs έχουν δυσκολία να εντοπίσουν με ακρίβεια αντικείμενα ή περιοχές μέσα σε εικόνες. Παρόλα αυτά, όταν χρησιμοποιούν το UICrit dataset, μοντέλα όπως το Gemini-1.5-pro και το GPT-4o, και προσθέτουν οπτικά βοηθήματα, όπως σημάδια συντεταγμένων στα άκρα της εικόνας και μεγενθυμένες εικόνες γύρω από το προτεινόμενο πλαίσιο, το μοντέλο μπορεί να επανεξετάσει την περιοχή και να διορθώσει τη θέση του πλαισίου. Τα αποτελέσματα δείχνουν ότι η επαναλαμβανόμενη οπτική προώθηση βελτιώνει τόσο την ποιότητα των παραγόμενων κριτικών όσο και την ακρίβεια των πλαισίων σε σχέση με πιο απλές μεθόδους αναφοράς. [3]

Παλι όμως η συγκεκριμένη εργασία είναι προσανατολισμένη σε στατικά screenshots. Δεν ασχολείται με την καταγραφή διεπαφών μέσω κάμερας, ούτε με προβλήματα που εμφανίζονται όταν η εικόνα προέρχεται από πραγματική οθόνη (σαν γωνία λήψης, αντανάκλαση, θόλωση, moiré patterns κ.λπ.). Επίσης, δεν ενσωματώνει AR ή Web AR αλληλεπίδραση. Τα bounding boxes παράγονται πάνω στο screenshot ως αποτέλεσμα ανάλυσης, ενώ στην παρούσα εφαρμογή τα αποτελέσματα προβάλλονται ως AR-style στοιχεία πάνω στην ορατή διεπαφή, ώστε ο χρήστης να συνδέει άμεσα το πρόβλημα με το αντίστοιχο σημείο της οθόνης.

2.3 Δυνατά και Αδύναμα Σημεία Ανταγωνιστικών Λύσεων

Οι ακαδημαϊκές εργασίες που αναφέρθηκαν έχουν κάποια πολύ σημαντικά δυνατά σημεία. Το πρώτο από αυτά είναι ότι δείχνουν ότι τα μοντέλα της τεχνητής νοημοσύνης μπορούν να χρησιμοποιηθούν για να κάνουν κριτική σε σχέση με την εμπειρία του χρήστη και το design μιας διεπαφής, εφόσον προστατευθούν με τα κατάλληλα δεδομένα, τις σωστές οδηγίες και παραδείγματα. Το δεύτερο σημείο είναι ότι συνδέουν τα σχόλια για την κριτική με συγκεκριμένες περιοχές της διεπαφής, χρησιμοποιώντας τετράγωνα για να περιγράψουν τις περιοχές, κάτι που κάνει τα σχόλια πιο κατανοητά και πιο εύκολα να εφαρμοστούν. Το τρίτο και τελευταίο σημείο είναι ότι δείχνουν ότι η αυτοματοποιημένη αξιολόγηση δεν πρέπει να περιορίζεται σε γενικά σχόλια, αλλά πρέπει να βασίζεται σε οπτικές παρατηρήσεις, να ταξινομεί τα προβλήματα και να δίνει πρακτικές προτάσεις για βελτίωση.

Οι συγκεκριμένες εργασίες έχουν ορισμένους περιορισμούς. Οι περισσότερες εργασίες βασίζονται σε ήδη διαθέσιμες εικόνες ή σε εικόνες από υπάρχοντα datasets. Αυτό σημαίνει ότι δεν εξετάζουν σε βάθος τις δυσκολίες που προκύπτουν όταν η είσοδος προέρχεται απευθείας από μια κάμερα. Υπάρχουν δυσκολίες όπως η παραμόρφωση της προοπτικής, οι αλλαγές στον φωτισμό, η θόλωση λόγω κίνησης και ο δύσκολος εντοπισμός μικρών στοιχείων της διεπαφής του χρήστη. Οι προσεγγίσεις αυτές λειτουργούν κυρίως ως ερευνητικά συστήματα. Δεν είναι ολοκληρωμένα εργαλεία που θα μπορούσαν να χρησιμοποιηθούν άμεσα από έναν designer ή developer σε πραγματικές συνθήκες. Για παράδειγμα, δεν μπορούν να χρησιμοποιηθούν στο πλήρες workflow μιας εφαρμογής που αξιοποιεί την κάμερα ενός κινητού τηλεφώνου.

Στην αγορά υπάρχουν εργαλεία που μας βοηθούν να ελέγξουμε την προσβασιμότητα των ιστοσελίδων. Το WAVE και το axe DevTools είναι δύο τέτοια εργαλεία. Αυτά τα εργαλεία μπορούν να εξετάσουν τη δομή μιας ιστοσελίδας και να δουν αν είναι προσβάσιμη για όλους τους χρήστες. Για να χρησιμοποιήσουμε αυτά τα εργαλεία, πρέπει να έχουμε πρόσβαση στη σελίδα μέσω του browser ή των developer tools. Αυτό σημαίνει ότι αυτά τα εργαλεία δεν είναι κατάλληλα για να αξιολογήσουμε μια διεπαφή που προβάλλεται σε μια εξωτερική οθόνη. Υπάρχουν και άλλα εργαλεία, όπως το Attention Insight, που χρησιμοποιούν τη τεχνητή νοημοσύνη για να δημιουργήσουν heatmaps και να εκτιμήσουν την οπτική προσοχή των χρηστών. Αυτά τα εργαλεία μας δείχνουν πού στρέφεται η προσοχή των χρηστών όταν βλέπουν μια σελίδα. Ωστόσο, αυτά τα εργαλεία έχουν κάποιους περιορισμούς. Δεν καλύπτουν όλες τις πτυχές της αξιολόγησης μιας διεπαφής. Για παράδειγμα, δεν μας επιτρέπουν να δούμε σχόλια μέσω της augmented reality ή να δημιουργήσουμε μια οργανωμένη αναφορά. [5], [6], [7]

Με βάση τις δύο εργασίες που εξετάστηκαν, φαίνεται ότι καθεμία καλύπτει διαφορετικό μέρος του προβλήματος. Η πρώτη επικεντρώνεται κυρίως στην παραγωγή σχολίων για διεπαφές μέσα από ερευνητικά datasets, ενώ η δεύτερη εξετάζει πιο συγκεκριμένα στοιχεία της αξιολόγησης. Παρ' όλα αυτά, καμία από τις δύο δεν υποστηρίζει μια ολοκληρωμένη διαδικασία που να ξεκινά από τη λήψη της διεπαφής με κάμερα και να καταλήγει στον εντοπισμό και την οπτική παρουσίαση των προβλημάτων πάνω στην ίδια την οθόνη.

2.4 Κενά της Αγοράς και Καινοτομία της Προτεινόμενης Εφαρμογής

Από τη μελέτη των σχετικών εργασιών και εργαλείων διαπιστώνεται ότι κάθε προσέγγιση εξετάζει διαφορετικό μέρος της αξιολόγησης διεπαφών. Κάποιες λύσεις δίνουν έμφαση σε ζητήματα προσβασιμότητας, αξιοποιώντας πληροφορίες από το DOM, ενώ άλλες βασίζονται σε στατικές εικόνες της διεπαφής για την παραγωγή σχολίων σχεδιασμού ή για την εκτίμηση των σημείων που είναι πιθανότερο να προσελκύσουν την προσοχή του χρήστη. Με βάση τις λύσεις που εξετάστηκαν, δεν εντοπίστηκε κάποια προσέγγιση που να συνδυάζει τη λήψη μιας διεπαφής με κάμερα, την αξιολόγησή της με τεχνικές τεχνητής νοημοσύνης και την οπτική τοποθέτηση των ευρημάτων επάνω στην εικόνα μέσω overlays παρόμοιων με αυτά της επαυξημένης πραγματικότητας. Στο συγκεκριμένο κενό επιχειρεί να ανταποκριθεί η προτεινόμενη εφαρμογή.

2.4.1 Κενά της Αγοράς

Από την εξέταση των εργαλείων και των ερευνητικών εργασιών που παρουσιάστηκαν, προκύπτει ότι η αξιολόγηση μιας web διεπαφής συνήθως προϋποθέτει κάποια μορφή άμεσης πρόσβασης στο ψηφιακό της περιβάλλον. Σε αρκετές περιπτώσεις απαιτείται η εισαγωγή ενός URL, η χρήση browser extension ή developer tools, η μεταφόρτωση ενός screenshot ή η πρόσβαση στο αρχείο σχεδίασης. Οι απαιτήσεις αυτές δεν εξυπηρετούν πάντοτε περιπτώσεις στις οποίες ο χρήστης θέλει να εξετάσει γρήγορα μια διεπαφή όπως ακριβώς εμφανίζεται σε μια οθόνη, για παράδειγμα σε laptop, tablet ή δημόσιο display.

Εργαλεία προσβασιμότητας, όπως το WAVE και το axe DevTools, είναι κατάλληλα για τον εντοπισμό τεχνικών προβλημάτων, καθώς αξιοποιούν τη δομή της ιστοσελίδας και τις πληροφορίες του DOM. Ωστόσο, η λειτουργία τους βασίζεται στην πρόσβαση στη σελίδα μέσω browser. Από την άλλη πλευρά, οι εργασίες UICrit και Visual Prompting with Iterative Refinement for Design Critique Generation μελετούν την παραγωγή σχολίων σχεδιασμού από στατικά screenshots. Παρότι συνδέουν ορισμένα ευρήματα με συγκεκριμένες περιοχές της εικόνας μέσω bounding boxes, δεν επικεντρώνονται στην

περίπτωση όπου η εικόνα της διεπαφής λαμβάνεται απευθείας από την κάμερα ενός κινητού, κάτω από πραγματικές συνθήκες χρήσης.

Ένα επιπλέον ζήτημα αφορά τον τρόπο με τον οποίο παρουσιάζονται τα αποτελέσματα της αξιολόγησης. Οι λύσεις που εξετάστηκαν χρησιμοποιούν κυρίως λίστες παρατηρήσεων, αναφορές, επισημάνσεις πάνω σε screenshots ή heatmaps. Η σύνδεση, όμως, ανάμεσα σε ένα εύρημα και στο αντίστοιχο σημείο της οθόνης δεν είναι πάντοτε άμεση. Έτσι, ο χρήστης ενδέχεται να χρειάζεται να εντοπίσει μόνος του ποιο στοιχείο της διεπαφής αφορά κάθε παρατήρηση, κάτι που μπορεί να δυσκολέψει την κατανόηση των αποτελεσμάτων, ιδιαίτερα όταν η διεπαφή περιλαμβάνει πολλά ή πυκνά διατεταγμένα στοιχεία.

Με βάση τα παραπάνω, το κενό που επιχειρεί να καλύψει η προτεινόμενη εφαρμογή δεν αφορά απλώς τη χρήση τεχνητής νοημοσύνης στην αξιολόγηση UI/UX. Αφορά κυρίως τον συνδυασμό της λήψης εικόνας μέσω κάμερας, της αυτόματης παραγωγής σχολίων, του εντοπισμού των σχετικών προβλημάτων πάνω στη διεπαφή και της οπτικής παρουσίας τους με επισημάνσεις που παραπέμπουν σε περιβάλλον επαυξημένης πραγματικότητας.

2.4.2 Καινοτομία της Προτεινόμενης Εφαρμογής

Η προτεινόμενη εφαρμογή ακολουθεί έναν διαφορετικό τρόπο αλληλεπίδρασης με τον χρήστη. Για την αξιολόγηση μιας διεπαφής δεν απαιτείται η εγκατάσταση πρόσθετου στον browser, η πρόσβαση στον κώδικα της ιστοσελίδας ή η χειροκίνητη μεταφόρτωση ενός screenshot. Ο χρήστης ανοίγει την εφαρμογή στο κινητό του, στρέφει την κάμερα προς την οθόνη που θέλει να εξετάσει και καταγράφει τη διεπαφή. Η εικόνα αναλύεται από το σύστημα τεχνητής νοημοσύνης και τα προβλήματα που εντοπίζονται επιστρέφονται σε δομημένη μορφή. Στη συνέχεια, προβάλλονται πάνω στην εικόνα ως οπτικές επισημάνσεις, ώστε να γίνεται άμεσα αντιληπτό σε ποιο σημείο αναφέρεται κάθε παρατήρηση. Η ιδιαιτερότητα της εφαρμογής δεν περιορίζεται στη χρήση τεχνητής νοημοσύνης, αλλά αφορά κυρίως τον συνδυασμό διαφορετικών τεχνολογιών σε μία ενιαία διαδικασία. Η υπολογιστική όραση εντοπίζει στοιχεία της διεπαφής και τη θέση τους στην εικόνα. Τα δεδομένα αυτά συνοδεύουν την αξιολόγηση UI/UX, η οποία επιστρέφεται σε μορφή JSON. Η αξιολόγηση αρχίζει από την εικόνα της οθόνης που καταγράφει ο χρήστης με την κάμερα.

2.5 Επίλογος

Η ανασκόπηση καλύπτει εργαλεία ελέγχου κώδικα και υπηρεσίες οπτικής αξιολόγησης. Περιλαμβάνει επίσης ερευνητικά συστήματα που παράγουν κριτική για διεπαφές χρήστη. Καμία από αυτές τις προσεγγίσεις δεν συνδυάζει από μόνη της τη ζωντανή παρατήρηση μιας οθόνης με τη χωρική σύνδεση των ευρημάτων στην εικόνα.

Στο ARtique, το YOLO εντοπίζει τα ορατά στοιχεία που χρησιμοποιούνται ως σημεία αγκύρωσης για τις παρατηρήσεις του Gemini. Τα ευρήματα προβάλλονται στο περιβάλλον WebAR. Η μεθοδολογία της εφαρμογής και οι περιορισμοί της περιγράφονται στο επόμενο κεφάλαιο.

Κεφάλαιο 3: Μεθοδολογία και Σχεδιαστικές Αποφάσεις

3.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφονται οι τεχνολογικές και σχεδιαστικές επιλογές της υλοποίησης του ARtique. Η περιγραφή ακολουθεί τη ροή από τη λήψη της εικόνας έως την αποθήκευση του audit.

Οι επιλογές εξετάζονται και ως προς τους περιορισμούς τους. Το YOLO εντοπίζει τα ορατά στοιχεία της διεπαφής, ενώ το Gemini αξιολογεί τα ζητήματα UI/UX. Η ποιότητα της λήψης επηρεάζει τη χωρική προβολή των ευρημάτων.

Το κεφάλαιο παρουσιάζει τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Μετά τον καθορισμό των λειτουργικών απαιτήσεων, μελετήθηκαν η αρχιτεκτονική και οι τεχνολογίες του συστήματος.

Κατά την επιλογή των τεχνολογιών ελέγχθηκαν οι άδειες χρήσης και το κόστος που θα μπορούσε να προκύψει. Προτιμήθηκαν λύσεις ανοιχτού κώδικα στις περιπτώσεις όπου κάλυπταν τις ανάγκες της υλοποίησης. Για την εκπαίδευση του μοντέλου ανίχνευσης χρησιμοποιήθηκαν datasets διαθέσιμα χωρίς κόστος άδειας, ενώ η εκπαίδευση έγινε εκτός της τελικής εφαρμογής.

Παράλληλα εξετάστηκε αν οι τεχνολογίες μπορούσαν να συνεργαστούν μέσα στο ίδιο web περιβάλλον. Αυτό ήταν απαραίτητο, επειδή η εφαρμογή συνδυάζει camera input, ανίχνευση UI elements, επικοινωνία με backend και αποθήκευση των αποτελεσμάτων.

3.2 Επιλογή Γλωσσών Προγραμματισμού

Για την ανάπτυξη του ARtique χρησιμοποιήθηκαν διαφορετικές γλώσσες, επειδή η εφαρμογή δεν περιορίζεται σε ένα απλό frontend. Περιλαμβάνει περιβάλλον χρήστη στον browser, πρόσβαση στην κάμερα, ανίχνευση στοιχείων πάνω στην εικόνα, επικοινωνία με backend υπηρεσία, αποθήκευση audits και ξεχωριστή διαδικασία εκπαίδευσης του μοντέλου YOLO. Έτσι, η επιλογή των γλωσσών έγινε ανάλογα με τον ρόλο κάθε μέρους της εφαρμογής.

Στο frontend επιλέχθηκε η TypeScript, καθώς χρησιμοποιείται από το Angular Framework και βοηθά στην οργάνωση μεγαλύτερων εφαρμογών. Στο ARtique υπάρχουν αρκετά δεδομένα με συγκεκριμένη δομή, όπως τα αποτελέσματα του audit, τα bounding boxes, οι κατηγορίες των UI/UX issues και τα UI elements που εντοπίζονται από το YOLO. Οι τύποι και τα interfaces βοηθούν στον έλεγχο αυτών των δεδομένων μέσα στην εφαρμογή.

Για το backend χρησιμοποιήθηκε JavaScript σε περιβάλλον Node.js με ExpressJS. Το backend δέχεται την εικόνα από το frontend, την προετοιμάζει, τη στέλνει στο μοντέλο τεχνητής νοημοσύνης και αποθηκεύει το τελικό αποτέλεσμα στη βάση δεδομένων. Με αυτή την επιλογή η backend πλευρά παραμένει απλή και κοντά στις ανάγκες του πρωτοτύπου.

Για το κομμάτι της μηχανικής μάθησης χρησιμοποιήθηκε Python. Η γλώσσα αυτή χρησιμοποιείται συχνά για προετοιμασία dataset, μετατροπή annotations, εκπαίδευση YOLO μοντέλων και εξαγωγή τους σε μορφή ONNX.

3.2.1 TypeScript

Η TypeScript χρησιμοποιήθηκε στο frontend της εφαρμογής, καθώς αποτελεί τη βασική γλώσσα ανάπτυξης στο Angular. Πρόκειται για γλώσσα που βασίζεται στη JavaScript, αλλά προσθέτει στατική τυποποίηση. Αυτό ήταν χρήσιμο στο ARtique, επειδή η εφαρμογή χειρίζεται αρκετές δομές δεδομένων που πρέπει να παραμένουν σταθερές σε όλη τη ροή της λειτουργίας της. [8]

Για παράδειγμα, τα αποτελέσματα που επιστρέφονται από την AI ανάλυση δεν είναι απλό κείμενο. Περιλαμβάνουν πεδία όπως κατηγορία προβλήματος, σοβαρότητα, βαθμό εμπιστοσύνης, προτεινόμενη βελτίωση και bounding box. Αν αυτά τα δεδομένα δεν έχουν καθαρά ορισμένη μορφή, είναι εύκολο να προκύψουν σφάλματα κατά την εμφάνιση των αποτελεσμάτων ή κατά την τοποθέτηση των οπτικών επισημάνσεων πάνω στην εικόνα.

Στην εφαρμογή ορίστηκαν τύποι για τα δεδομένα του audit και για τα στοιχεία που εντοπίζει το YOLO. Ορίστηκαν επίσης τύποι για τα δεδομένα της κάμερας. Τα interfaces περιγράφουν τα δεδομένα που δέχονται και επιστρέφουν τα components και τα services.

Η TypeScript διευκόλυνε τη συντήρηση του κώδικα όταν άλλαζαν οι δομές των δεδομένων. Το frontend καλύπτει τη ροή από την προεπισκόπηση της κάμερας έως την εμφάνιση του audit. Η ροή περιλαμβάνει την ανίχνευση της οθόνης και την αναγνώριση στοιχείων με YOLO. Περιλαμβάνει ακόμη τις οπτικές επισημάνσεις και το ιστορικό των αξιολογήσεων. Η TypeScript βοηθά στην παρακολούθηση των δεδομένων όταν περνούν από ένα στάδιο της ροής στο επόμενο.

Το Angular χρησιμοποιεί TypeScript στα components και στα services του. Η ίδια γλώσσα χρησιμοποιείται και για το routing και τα models. Η TypeScript επιτρέπει να ορίζονται οι τύποι δίπλα στον κώδικα των Angular components. Στο Σχήμα 3.1 εμφανίζεται το επίσημο λογότυπο της TypeScript.



Σχήμα 3.1: Επίσημο λογότυπο της TypeScript.

Πηγή: TypeScript Branding [9].

Για το backend της εφαρμογής επέλεξα JavaScript με Node.js. Το backend στο ARtique δεν είναι ένα ξεχωριστό μεγάλο σύστημα με πολλές πολύπλοκες λειτουργίες. Ο βασικός του ρόλος είναι να δέχεται την εικόνα που στέλνει το frontend, να παραλαμβάνει μαζί και τα στοιχεία που έχουν εντοπιστεί από το YOLO, να καλεί την υπηρεσία τεχνητής νοημοσύνης και μετά να αποθηκεύει το αποτέλεσμα στη βάση δεδομένων.

Το Node.js ήταν βολική επιλογή για αυτή τη δουλειά, επειδή χειρίζεται καλά λειτουργίες που γίνονται ασύγχρονα. Για παράδειγμα, όταν ο χρήστης πατάει για ανάλυση, το backend πρέπει να περιμένει την αποστολή της εικόνας, την απάντηση από το Gemini API και την εγγραφή στη MySQL. Αυτά τα βήματα δεν γίνονται όλα αμέσως, οπότε χρειαζόταν ένα περιβάλλον που να μπορεί να τα διαχειριστεί χωρίς να μπλοκάρει τη λειτουργία του server. [10]

Πάνω στο Node.js χρησιμοποιήθηκε το Express.js. Με το Express οργανώθηκαν τα βασικά API endpoints της εφαρμογής. Το πιο βασικό endpoint είναι αυτό που δέχεται την εικόνα για ανάλυση. Υπάρχουν επίσης endpoints για την ανάκτηση αποθηκευμένων audits και για την αποθήκευση feedback. Με αυτόν τον τρόπο το backend χωρίστηκε σε routes, controllers, services, models και helpers, ώστε κάθε αρχείο να έχει πιο συγκεκριμένο ρόλο. [11]

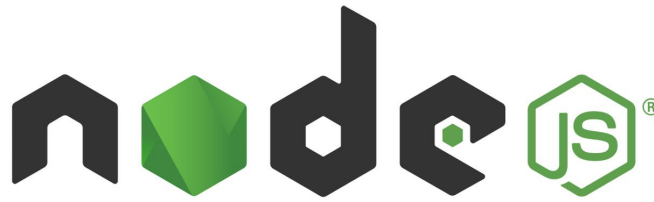
Στο backend δεν χρησιμοποίησα TypeScript, αλλά απλή JavaScript. Αυτό έγινε κυρίως για να μείνει η υλοποίηση πιο απλή. Επειδή το μεγαλύτερο μέρος της πολυπλοκότητας βρίσκεται στο frontend, στην κάμερα, στο screen detection, στο YOLO και στην AI ανάλυση, δεν ήθελα να προστεθεί επιπλέον βάρος στο backend με περισσότερη ρύθμιση εργαλείων. Στο backend δεν χρησιμοποίησα TypeScript, αλλά απλή JavaScript. Αυτό έγινε κυρίως για να μείνει η υλοποίηση πιο απλή. Το μεγαλύτερο μέρος της επεξεργασίας εκτελείται στο frontend. Η πρόσβαση στην κάμερα και ο εντοπισμός της οθόνης γίνονται στον browser. Το YOLO εκτελείται επίσης εκεί. Η JavaScript κάλυπτε τις ανάγκες του backend χωρίς πρόσθετη ρύθμιση εργαλείων.

Το backend δέχεται την εικόνα και τα δεδομένα των στοιχείων που έχει εντοπίσει το YOLO. Ζητά την αξιολόγηση από το Gemini και αποθηκεύει το audit στη MySQL. Το αποτέλεσμα επιστρέφεται στο Angular για προβολή στην αναφορά και στα AR overlays.

3.2.2 JavaScript και Nodejs

Για το backend της εφαρμογής επέλεξα JavaScript με Node.js. Το backend στο ARtique έχει περιορισμένο ρόλο. Δέχεται την εικόνα που στέλνει το frontend μαζί με τα στοιχεία που έχει εντοπίσει το YOLO. Στη συνέχεια καλεί το Gemini και αποθηκεύει το audit στη βάση δεδομένων.

Το Node.js επιλέχθηκε επειδή το backend εκτελεί ασύγχρονες λειτουργίες. Μετά το αίτημα ανάλυσης, περιμένει την εικόνα από το frontend και την απόκριση του Gemini API. Στη συνέχεια καταγράφει το audit στη MySQL. Η ροή αυτή δεν πρέπει να δεσμεύει τη λειτουργία του server. [10] Στο Σχήμα 3.2 παρουσιάζεται επίσης μορφή του Node.js.



Σχήμα 3.2: Επίσημο λογότυπο του Node.js.

Πηγή: Node.js Branding [12].

Πάνω στο Node.js χρησιμοποιήθηκε το Express.js. Με το Express οργανώθηκαν τα βασικά API endpoints της εφαρμογής. Το πιο βασικό endpoint είναι αυτό που δέχεται την εικόνα για ανάλυση. Υπάρχουν επίσης endpoints για την ανάκτηση αποθηκευμένων audits και για την αποθήκευση feedback. Με αυτόν τον τρόπο το backend χωρίστηκε σε routes, controllers, services, models και helpers, ώστε κάθε αρχείο να έχει πιο συγκεκριμένο ρόλο. [11] Στο Σχήμα 3.3 παρουσιάζεται επίσης το λογότυπο του Express.js.



Σχήμα 3.3: Επίσημο λογότυπο του Express.js.

Πηγή: Express.js Community [13].

Στο backend δεν χρησιμοποίησα TypeScript, αλλά απλή JavaScript. Αυτό έγινε κυρίως για να μείνει η υλοποίηση πιο απλή. Επειδή το μεγαλύτερο μέρος της πολυπλοκότητας βρίσκεται στο frontend, στην κάμερα, στο screen detection, στο YOLO και στην AI ανάλυση, δεν ήθελα να προστεθεί επιπλέον βάρος στο backend με περισσότερη ρύθμιση εργαλείων. Για τις ανάγκες της διπλωματικής, η plain JavaScript ήταν αρκετή.

Έτσι, το backend λειτουργεί σαν το ενδιάμεσο κομμάτι της εφαρμογής. Παίρνει τα δεδομένα από το Angular frontend, τα προετοιμάζει, επικοινωνεί με το Gemini, αποθηκεύει το audit στη MySQL και επιστρέφει στο frontend ένα αποτέλεσμα που μπορεί να εμφανιστεί στην αναφορά και στα AR overlays.

3.2.3 Python

Η Python χρησιμοποιήθηκε στο κομμάτι της μηχανικής μάθησης και όχι στο κύριο frontend ή backend της εφαρμογής. Συγκεκριμένα, χρησιμοποιήθηκε για την προετοιμασία των δεδομένων, τη μετατροπή των annotations σε μορφή YOLO, την εκπαίδευση του μοντέλου και την εξαγωγή του σε μορφή ONNX, ώστε να μπορεί αργότερα να τρέξει μέσα στο web περιβάλλον της εφαρμογής.

Ο λόγος που επιλέχθηκε η Python είναι ότι τα περισσότερα εργαλεία για computer vision και training μοντέλων είναι ήδη διαθέσιμα σε αυτή τη γλώσσα. Βιβλιοθήκες όπως το Ultralytics, το OpenCV, το NumPy και το ONNX Runtime χρησιμοποιούνται συχνά για τέτοιες εργασίες και μειώνουν αρκετά τον χρόνο που χρειάζεται για να στηθεί ένα πειραματικό pipeline. [14], [15], [16]

Στην περίπτωση του ARTique, η Python δεν χρησιμοποιήθηκε για να κάνει την τελική αξιολόγηση της διεπαφής. Αυτό είναι σημαντικό να ξεκαθαριστεί. Το YOLO μοντέλο που εκπαιδεύτηκε με τη βοήθεια της Python έχει ως ρόλο να εντοπίζει οπτικά στοιχεία της ιστοσελίδας, όπως κουμπιά, εικόνες, κείμενα ή περιοχές περιεχομένου. Η αξιολόγηση UI/UX γίνεται αργότερα από το μοντέλο τεχνητής νοημοσύνης, το οποίο λαμβάνει την εικόνα και τα στοιχεία που έχουν εντοπιστεί.

Ένα επιπλέον κομμάτι όπου βοήθησε η Python ήταν η προσαρμογή των datasets στο περιβάλλον της κάμερας. Τα περισσότερα διαθέσιμα datasets περιέχουν καθαρά screenshots, ενώ η δική μου εφαρμογή βλέπει την ιστοσελίδα μέσα από κάμερα κινητού που κοιτάζει μια οθόνη. Αυτό δημιουργεί παραμορφώσεις, θόρυβο, αντανakλάσεις και αλλαγές στη φωτεινότητα. Για αυτό δημιουργήθηκε διαδικασία augmentation, ώστε οι εικόνες εκπαίδευσης να μοιάζουν περισσότερο με αυτό που βλέπει πραγματικά η εφαρμογή.

Μετά την εκπαίδευση, το μοντέλο εξάγεται σε μορφή ONNX. Αυτή η μορφή επιτρέπει στο μοντέλο να χρησιμοποιηθεί από το frontend μέσω του ONNX Runtime Web. Έτσι, η Python χρησιμοποιείται κυρίως στο στάδιο της προετοιμασίας και εκπαίδευσης, ενώ η εκτέλεση του μοντέλου στην τελική εφαρμογή γίνεται μέσα στον browser. [15] Στο Σχήμα 3.4 παρουσιάζεται επίσημο λογότυπο της Python.



Σχήμα 3.4: Επίσημο λογότυπο της Python.

Πηγή: Python Software Foundation [17].

3.2.4 JSON και SQL για ανταλλαγή και αποθήκευση δεδομένων

Πέρα από τις γλώσσες προγραμματισμού, στην εφαρμογή χρησιμοποιήθηκαν το JSON και η SQL για τη μεταφορά και την αποθήκευση δεδομένων. Το JSON είναι μορφή ανταλλαγής δεδομένων, ενώ η SQL χρησιμοποιείται για την επικοινωνία με τη βάση. Και τα δύο ήταν απαραίτητα για τη λειτουργία του συστήματος. [18], [19]

Το JSON χρησιμοποιήθηκε κυρίως για να μεταφερθούν δεδομένα ανάμεσα στο πρόγραμμα που βλέπει ο χρήστης, το πίσω μέρος της εφαρμογής και το μοντέλο τεχνητής νοημοσύνης. Η απάντηση που δίνει το μοντέλο δεν είναι απλά κείμενο με σχόλια. Είναι δομημένη πληροφορία. Για κάθε πρόβλημα

υπάρχουν πεδία όπως τίτλος, κατηγορία, σοβαρότητα, βαθμός εμπιστοσύνης, περιγραφή, πρόταση για βελτίωση και όταν υπάρχει, τα σημεία του bounding box.

Αυτή η μορφή ήταν σημαντική για την εφαρμογή. Γιατί το πρόγραμμα πρέπει να ξέρει πώς να δείξει τα αποτελέσματα. Για παράδειγμα, η λίστα προβλημάτων χρειάζεται τα κείμενα και τις κατηγορίες. Τα AR overlays χρειάζονται τις συντεταγμένες του bounding box ή το στοιχείο πάνω στο οποίο έχει αγκυρωθεί το πρόβλημα. Αν η απάντηση ήταν απλό κείμενο, θα ήταν πολύ πιο δύσκολο να δεις ποιο πρόβλημα αντιστοιχεί σε ποιο σημείο της οθόνης.

Το SQL χρησιμοποιήθηκε για να αποθηκεύσουμε τα αποτελέσματα σε μια βάση δεδομένων. Στην εφαρμογή υπάρχουν πίνακες για τα audits, για τα προβλήματα που βρέθηκαν και για το feedback που μπορεί να δοθεί. Με αυτόν τον τρόπο, κάθε ανάλυση που γίνεται δεν χάνεται μετά το τέλος. Μπορεί να ανακτηθεί όποτε χρειαστεί. [19]

Η χρήση μιας βάσης δεδομένων ήταν χρήσιμη και για τη διπλωματική, όχι μόνο για την ίδια την εφαρμογή. Τα αποθηκευμένα audits μπορούν να χρησιμοποιηθούν για να συγκριθούν αποτελέσματα, για να ελεγχθεί πόσο ακριβή είναι οι προτάσεις ή για να γίνει αξιολόγηση από χρήστες ή επιθεωρητές. Γι' αυτό το λόγο, το SQL δεν ήταν μόνο για να μείνει ιστορικό. Υπήρχε για να βοηθήσει και στην αξιολόγηση του συστήματος στο μέλλον.

Έτσι, το JSON χρησιμοποιείται για να μεταφέρεται το αποτέλεσμα σε μια δομημένη μορφή, ενώ το SQL χρησιμοποιείται για να αποθηκεύεται και να επαναφέρεται. Τα δύο αυτά μέρη συνδέουν την ανάλυση της τεχνητής νοημοσύνης με την πρακτική χρήση της εφαρμογής.

3.3 Κριτήρια Επιλογής Τεχνολογιών

Στην επιλογή των τεχνολογιών για το ARtique προσπάθησα πρώτα να ξεκινήσω από το ίδιο το πρόβλημα της εφαρμογής. Ο χρήστης δεν ανεβάζει απλώς ένα screenshot από τον υπολογιστή του. Ανοίγει την εφαρμογή, χρησιμοποιεί την κάμερα και στρέφει τη συσκευή του προς μια οθόνη όπου εμφανίζεται μια ιστοσελίδα. Αυτό σημαίνει ότι οι τεχνολογίες έπρεπε να μπορούν να δουλέψουν με camera feed, εικόνα, overlays και επικοινωνία με backend.

Για το frontend χρειαζόταν ένα framework που να μπορεί να οργανώσει αρκετές οθόνες και υπηρεσίες χωρίς να γίνει ο κώδικας δύσκολος στη συντήρηση. Για αυτό χρησιμοποιήθηκε το Angular. Στην εφαρμογή υπάρχουν διαφορετικά κομμάτια, όπως η αρχική σελίδα, η οθόνη σάρωσης, η προεπισκόπηση της εικόνας, η αναφορά αποτελεσμάτων και το ιστορικό των audits. Το Angular βοήθησε να χωριστούν αυτά τα μέρη σε components και services. [20] Στο Σχήμα 3.5 παρουσιάζεται επίσημο λογότυπο του Angular.



Σχήμα 3.5: Επίσημο λογότυπο του Angular.

Πηγή: *Angular Press Kit [21]*.

Ένα άλλο κριτήριο ήταν η πρόσβαση στην κάμερα μέσα από browser. Η εφαρμογή έπρεπε να μπορεί να λειτουργεί χωρίς να εγκατασταθεί native εφαρμογή στο κινητό. Για αυτό χρησιμοποιήθηκαν browser APIs για την κάμερα και canvas για την επεξεργασία της εικόνας. Αυτή η επιλογή ταίριαζε περισσότερο με τον στόχο της εργασίας, επειδή το project αφορά Web AR και όχι μια κλασική Android εφαρμογή. [22], [23]

Για την ανίχνευση της οθόνης και των στοιχείων της διεπαφής χρειάστηκαν εργαλεία computer vision. Το OpenCV.js χρησιμοποιήθηκε για να εντοπίζεται η οθόνη μέσα στο camera feed. Στη συνέχεια, το YOLO μοντέλο χρησιμοποιήθηκε για να βρίσκει στοιχεία της ιστοσελίδας, όπως κουμπιά, περιοχές κειμένου και εικόνες. Επειδή το μοντέλο έπρεπε να τρέχει μέσα στον browser, έγινε εξαγωγή του σε μορφή ONNX και χρησιμοποιήθηκε το ONNX Runtime Web. [24], [14], [15], [16] Στο Σχήμα 3.6 παρουσιάζεται επίσημο λογότυπο του OpenCV, του οποίου η έκδοση OpenCV.js χρησιμοποιείται για screen detection. Στο Σχήμα 3.7 παρουσιάζεται επίσημο λογότυπο του Ultralytics YOLOv8 που χρησιμοποιήθηκε για την εκπαίδευση του UI-element detector. Στο Σχήμα 3.8 παρουσιάζεται επίσημο λογότυπο του ONNX Runtime, μέσω του οποίου εκτελείται το μοντέλο στον browser.



Σχήμα 3.6: Επίσημο λογότυπο του OpenCV, του οποίου η έκδοση OpenCV.js χρησιμοποιείται για screen detection.

Πηγή: *OpenCV Media Kit [25]*.



Σχήμα 3.7: Επίσημο λογότυπο του Ultralytics YOLOv8 που χρησιμοποιήθηκε για την εκπαίδευση του UI-element detector.

Πηγή: *Ultralytics Brand [26]*.



Σχήμα 3.8: Επίσημο λογότυπο του ONNX Runtime, μέσω του οποίου εκτελείται το μοντέλο στον browser.

Πηγή: ONNX Runtime [27].

Στο backend το ζητούμενο ήταν πιο πρακτικό. Έπρεπε να υπάρχει ένας server που να δέχεται την εικόνα, να παίρνει μαζί τα στοιχεία που εντόπισε το YOLO, να καλεί το Gemini και να αποθηκεύει το αποτέλεσμα. Για αυτό επιλέχθηκε Node.js με Express. Δεν χρειάστηκε ένα πιο βαρύ backend framework, επειδή οι βασικές λειτουργίες ήταν συγκεκριμένες: upload εικόνας, AI ανάλυση, αποθήκευση και ανάκτηση audits. Στο Σχήμα 3.9 παρουσιάζεται επίσημο οπτικό σήμα του Gemini API που χρησιμοποιείται για τη δομημένη UI/UX αξιολόγηση.



Σχήμα 3.9: Επίσημο οπτικό σήμα του Gemini API που χρησιμοποιείται για τη δομημένη UI/UX αξιολόγηση.

Πηγή: Google AI for Developers [28].

Το κόστος ήταν επίσης σημαντικό. Επειδή η εφαρμογή αναπτύχθηκε για διπλωματική εργασία, προτιμήθηκαν εργαλεία που είναι δωρεάν ή ανοιχτού κώδικα, όπως Angular, Node.js, Express, OpenCV.js, YOLO και MySQL. Αυτό βοήθησε να γίνει ανάπτυξη και πειραματισμός χωρίς να εξαρτάται η εργασία από εμπορικές άδειες.

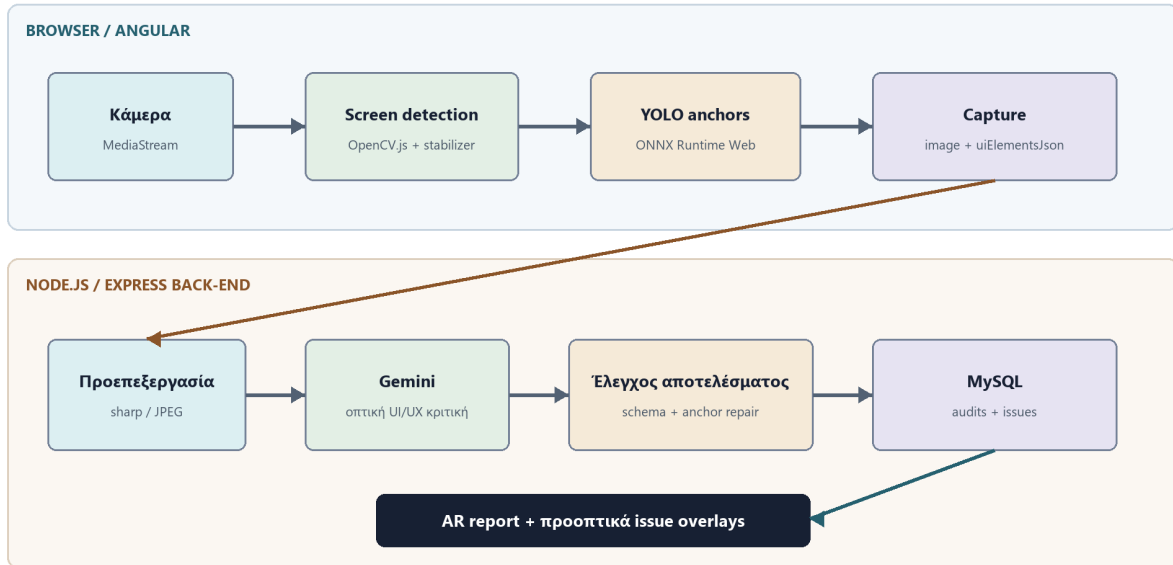
Τέλος, οι τεχνολογίες έπρεπε να επιτρέπουν και μελλοντική αξιολόγηση της εφαρμογής. Για αυτό τα αποτελέσματα της AI ανάλυσης επιστρέφονται σε JSON και αποθηκεύονται στη MySQL. Με αυτόν τον τρόπο, κάθε audit μπορεί να ανακτηθεί αργότερα και να συγκριθεί με ανθρώπινη αξιολόγηση ή με feedback χρηστών.

3.4 Συνολική Αρχιτεκτονική Συστήματος

Η εφαρμογή ARtique χωρίζεται σε δύο βασικά μέρη. Το πρώτο μέρος εκτελείται στο frontend, μέσα στον browser, και ασχολείται με την κάμερα, την ανίχνευση της οθόνης και την εμφάνιση των οπτικών επισημάνσεων. Το δεύτερο μέρος βρίσκεται στο backend και αναλαμβάνει την αποστολή της εικόνας στο μοντέλο τεχνητής νοημοσύνης, την επεξεργασία της απάντησης και την αποθήκευση των

αποτελεσμάτων. Στο Σχήμα 3.10 παρουσιάζεται συνολική αρχιτεκτονική και ροή δεδομένων της εφαρμογής ARtique.

ARtique: αρχιτεκτονική και ροή δεδομένων



Σχήμα 3.10: Συνολική αρχιτεκτονική και ροή δεδομένων της εφαρμογής ARtique.

Πηγή: Προσωπική υλοποίηση.

Στο frontend ο χρήστης ανοίγει την εφαρμογή και δίνει πρόσβαση στην κάμερα. Η κάμερα δείχνει μια οθόνη στην οποία εμφανίζεται μια ιστοσελίδα. Πριν γίνει η ανάλυση από το AI μοντέλο, η εφαρμογή προσπαθεί να εντοπίσει την ίδια την οθόνη μέσα στο camera feed. Για αυτό χρησιμοποιείται το OpenCV.js, ώστε να βρεθεί το πλαίσιο της οθόνης και να ξεχωρίσει η περιοχή που ενδιαφέρει την εφαρμογή.

Αφού εντοπιστεί η οθόνη, το σύστημα χρησιμοποιεί το YOLO μοντέλο για να αναγνωρίσει στοιχεία της διεπαφής μέσα σε αυτήν την περιοχή. Τέτοια στοιχεία μπορεί να είναι κουμπιά, εικόνες, περιοχές κειμένου ή άλλα ορατά μέρη της ιστοσελίδας. Το YOLO δεν αξιολογεί αν η διεπαφή είναι σωστή ή λάθος. Απλώς δίνει στην εφαρμογή πληροφορία για το πού βρίσκονται ορισμένα στοιχεία πάνω στην εικόνα.

Όταν ο χρήστης πατήσει για ανάλυση, το frontend κρατάει ένα στιγμιότυπο της εικόνας και το στέλνει στο backend. Μαζί με την εικόνα στέλνονται και τα στοιχεία που έχει εντοπίσει το YOLO εκείνη τη στιγμή. Αυτά τα δεδομένα βοηθούν το AI μοντέλο να συνδέσει τις παρατηρήσεις του με συγκεκριμένα σημεία της διεπαφής.

Το backend είναι υλοποιημένο με Node.js και Express. Παραλαμβάνει την εικόνα, την προετοιμάζει και καλεί το Gemini για να γίνει η UI/UX αξιολόγηση. Η απάντηση επιστρέφει σε μορφή JSON και περιλαμβάνει τη σύνοψη του audit, τα προβλήματα που βρέθηκαν, τη σοβαρότητα, την κατηγορία, την

πρόταση βελτίωσης και, όπου είναι δυνατό, τη σύνδεση με κάποιο στοιχείο που είχε εντοπιστεί από το YOLO.

Μετά την επεξεργασία της απάντησης, το backend αποθηκεύει το audit στη MySQL. Στη βάση κρατούνται τα βασικά στοιχεία της ανάλυσης, τα επιμέρους προβλήματα και προαιρετικά feedback για τα αποτελέσματα. Έτσι, ένα audit μπορεί να εμφανιστεί ξανά αργότερα και να χρησιμοποιηθεί και για την αξιολόγηση της διπλωματικής.

Στο τέλος, το frontend εμφανίζει τα αποτελέσματα στον χρήστη. Η αναφορά παρουσιάζει τα προβλήματα σε λίστα, ενώ τα AR overlays δείχνουν οπτικά σε ποιο σημείο της διεπαφής αναφέρεται κάθε παρατήρηση. Αυτός ο διαχωρισμός ήταν χρήσιμος στην υλοποίηση, επειδή το frontend κρατάει τη ζωντανή αλληλεπίδραση με την κάμερα, ενώ το backend κρατάει την AI ανάλυση και την αποθήκευση των δεδομένων.

3.5 Μηχανική Μάθηση και Υπολογιστική Όραση

Στην εφαρμογή ARtique η μηχανική μάθηση και η υπολογιστική όραση χρησιμοποιούνται κυρίως για να μπορέσει το σύστημα να καταλάβει πού βρίσκονται βασικά στοιχεία της ιστοσελίδας μέσα στην εικόνα της κάμερας. Δεν χρησιμοποιούνται για να αποφασίσουν από μόνες τους αν η διεπαφή είναι σωστή ή λάθος. Αυτό το κομμάτι γίνεται αργότερα από το μοντέλο τεχνητής νοημοσύνης που αναλαμβάνει την UI/UX αξιολόγηση. [24]

Το πρώτο βήμα είναι ο εντοπισμός της οθόνης μέσα στο camera feed. Επειδή ο χρήστης στρέφει το κινητό προς μια πραγματική οθόνη, η εικόνα περιέχει και άλλα στοιχεία του χώρου, όπως το γραφείο, τον τοίχο ή το πλαίσιο της οθόνης. Για αυτό χρησιμοποιήθηκε το OpenCV.js, ώστε η εφαρμογή να βρίσκει την περιοχή της εικόνας που αντιστοιχεί στην οθόνη και να μην επεξεργάζεται άσχετα σημεία.

Αφού εντοπιστεί η οθόνη, χρησιμοποιείται ένα YOLO μοντέλο για την ανίχνευση στοιχείων διεπαφής. Το μοντέλο τρέχει πάνω στο κομμάτι της εικόνας που αντιστοιχεί στην οθόνη και προσπαθεί να εντοπίσει στοιχεία όπως κουμπιά, εικόνες, περιοχές κειμένου και άλλα ορατά UI components. Τα αποτελέσματα αυτά επιστρέφονται ως bounding boxes, δηλαδή ορθογώνιες περιοχές που δείχνουν πού βρίσκεται κάθε στοιχείο. [24], [14]

Για να μπορούν τα bounding boxes να χρησιμοποιηθούν σε διαφορετικά μεγέθη εικόνας, οι συντεταγμένες τους κανονικοποιούνται. Δηλαδή δεν αποθηκεύονται μόνο ως απόλυτα pixels, αλλά ως

τιμές από 0 έως 1 σε σχέση με το πλάτος και το ύψος της εικόνας. Μια απλή μορφή αυτής της μετατροπής φαίνεται στη σχέση (3.1):

$$x_n = \frac{x_{(px)}}{W}, \quad y_n = \frac{y_{(px)}}{H}, \quad w_n = \frac{w_{(px)}}{W}, \quad h_n = \frac{h_{(px)}}{H} \quad (3.1)$$

όπου `W` είναι το πλάτος της εικόνας, `H` είναι το ύψος της εικόνας, ενώ `x\_px`, `y\_px`, `w\_px` και `h\_px` είναι οι αντίστοιχες τιμές του πλαισίου σε pixels. Με αυτόν τον τρόπο, το ίδιο bounding box μπορεί να προβληθεί σωστά ακόμα και αν η εικόνα εμφανιστεί σε διαφορετικό μέγεθος στην οθόνη του χρήστη.

Για την εκπαίδευση του YOLO μοντέλου χρησιμοποιήθηκε ξεχωριστό Python pipeline. Σε αυτό το στάδιο έγινε προετοιμασία των δεδομένων, μετατροπή των annotations στη μορφή που χρειάζεται το YOLO και εξαγωγή του τελικού μοντέλου σε μορφή ONNX. Η μορφή ONNX ήταν χρήσιμη επειδή επιτρέπει την εκτέλεση του μοντέλου μέσα στον browser μέσω του ONNX Runtime Web. [14], [15]

Ένα πρόβλημα που εμφανίστηκε είναι ότι τα περισσότερα datasets για UI elements βασίζονται σε καθαρά screenshots. Στην εφαρμογή όμως η ιστοσελίδα φαίνεται μέσα από κάμερα κινητού που κοιτάζει μια οθόνη. Αυτό σημαίνει ότι υπάρχουν αντανakλάσεις, θόρυβος, μικρή κλίση, θόλωση και αλλαγές στη φωτεινότητα. Για αυτό δημιουργήθηκε διαδικασία camera-domain augmentation, ώστε τα δεδομένα εκπαίδευσης να πλησιάζουν περισσότερο τις πραγματικές συνθήκες χρήσης.

Πρέπει επίσης να αναφερθεί ότι το YOLO μοντέλο στο ARtique δεν είναι το τελικό κριτήριο αξιολόγησης της διεπαφής. Ο ρόλος του είναι βοηθητικός. Δίνει στο σύστημα μια πρώτη χωρική πληροφορία, ώστε οι παρατηρήσεις του Gemini να μπορούν να συνδεθούν με συγκεκριμένα σημεία της οθόνης. Αν το YOLO δεν εντοπίσει σωστά κάποιο στοιχείο, τότε η τοποθέτηση των AR overlays μπορεί να γίνει λιγότερο ακριβής.

3.6 Backend και API Υποδομή

Στην ARtique το backend γράφτηκε σε Node.js με Express. Χρησιμοποιείται για τα κομμάτια που δεν είναι πρακτικό να γίνουν μέσα στον browser: παίρνει την εικόνα από το frontend, τη μετατρέπει σε κατάλληλη μορφή, καλεί το Gemini, αποθηκεύει το audit στη MySQL και επιστρέφει το JSON που περιμένει η Angular εφαρμογή.

Η διαδρομή της ανάλυσης περνά από το endpoint /api/audits/analyze. Όταν ο χρήστης πατήσει το κουμπί ανάλυσης, το frontend στέλνει ένα multipart/form-data request. Εκτός από την εικόνα της κάμερας, στο ίδιο request μπαίνουν και τα στοιχεία που είχαν εντοπιστεί εκείνη τη στιγμή από το live YOLO detection. Αυτά στέλνονται στο πεδίο uiElementsJson και περιγράφουν, για κάθε στοιχείο, το id, τον τύπο και το bounding box του πάνω στην εικόνα.

Στο backend, η εικόνα παραλαμβάνεται πρώτα από το multer, με memory storage. Εκεί γίνεται και ο πρώτος έλεγχος του αρχείου, ώστε να γίνονται δεκτές μόνο εικόνες PNG, JPEG ή WebP. Μετά

χρησιμοποιείται το `sharp` για την κανονικοποίηση της εικόνας. Στη `real` λειτουργία η εικόνα συμπιέζεται και, όταν χρειάζεται, μικραίνει πριν σταλεί στο Gemini, ώστε η κλήση να μη φορτώνεται με περιττά μεγάλο αρχείο. Η τελική εικόνα κρατιέται και ως JPEG αρχείο στον `server`, ενώ το αντίστοιχο `path` αποθηκεύεται στο `image_path` της εγγραφής του `audit`.

Η κλήση προς το Gemini δεν βρίσκεται στον Angular κώδικα. Γίνεται μέσα από το `AuditService` και το `AIService`, ώστε η λογική της ανάλυσης να μένει στο `backend`. Μαζί με την εικόνα, το Gemini λαμβάνει και τα `UI elements` από το `YOLO`. Έτσι, όταν γράφει ένα `issue`, μπορεί να το συνδέσει με κάποιο υπάρχον `elementId` αντί να υπολογίζει μόνο του από την αρχή τη θέση κάθε στοιχείου.

Πριν το αποτέλεσμα αποθηκευτεί και επιστραφεί στο `frontend`, το `backend` το περνά από κανονικοποίηση. Ελέγχονται οι κατηγορίες, το `severity`, το `checklist` και τα `bounding boxes`. Αν ένα `issue` δεν έχει σωστή σύνδεση με `YOLO element`, το `backend` προσπαθεί να βρει καλύτερη αντιστοίχιση με βάση την επικάλυψη ή την κοντινότερη θέση. Αυτό βοηθά ιδιαίτερα στην `AR` προβολή, όπου ένα λάθος `anchor` θα φαινόταν αμέσως ως πλαίσιο σε λάθος σημείο της οθόνης.

Στον Πίνακα 3.1 παρουσιάζονται τα βασικά `endpoints` που χρησιμοποιούνται στο `backend` της εφαρμογής. Ο Πίνακας 3.1 συνοψίζει βασικά `API endpoints` της εφαρμογής.

Πίνακας 3.1: Βασικά `API endpoints` της εφαρμογής

Μέθοδος	Endpoint	Περιγραφή
POST	/api/audits/analyze	Αποστολή εικόνας και <code>YOLO</code> στοιχείων για παραγωγή <code>UI/UX audit</code>
GET	/api/audits/:auditId	Ανάκτηση συγκεκριμένου <code>audit</code> με βάση το <code>public id</code>
GET	/api/audits	Ανάκτηση λίστας αποθηκευμένων <code>audits</code>
DELETE	/api/audits/:auditId	Διαγραφή <code>audit</code> από το ιστορικό
GET	/api/audits/:auditId/diagnostics	Προβολή διαγνωστικών δεδομένων σε <code>development mode</code>
POST	/api/audits/:auditId/feedback	Αποθήκευση <code>feedback</code> για τα αποτελέσματα του <code>audit</code>

Τα αποτελέσματα αποθηκεύονται σε `MySQL`. Στον πίνακα `audits` αποθηκεύονται το `public_id`, ο τύπος της πηγής, οι διαστάσεις της εικόνας, η περίληψη, οι περιορισμοί της ανάλυσης και το `raw AI response`. Τα επιμέρους προβλήματα μπαίνουν στον πίνακα `audit_issues`, όπου κάθε `issue` έχει δική του

κατηγορία, σοβαρότητα, τίτλο, περιγραφή, πρόταση βελτίωσης και, όταν υπάρχει, συντεταγμένες bounding box.

Για την αξιολόγηση της διπλωματικής υπάρχει και ο πίνακας audit_feedback. Δεν χρειάζεται για να δει ο χρήστης ένα audit, αλλά μπορεί να χρησιμοποιηθεί αργότερα από reviewer. Εκεί μπορεί να σημειωθεί αν μια παρατήρηση ήταν χρήσιμη, αν ήταν ακριβής ή αν η σοβαρότητά της θα έπρεπε να είναι διαφορετική.

3.7 Μεθοδολογία AI UI/UX Αξιολόγησης

Στο ARTique η αξιολόγηση της διεπαφής δεν εκτελείται από το Angular frontend. Το frontend αναλαμβάνει την κάμερα, την εμφάνιση της ζωντανής προβολής και τη συλλογή των στοιχείων που φαίνονται στην οθόνη. Η επικοινωνία με το Google Gemini γίνεται από το backend. Η επιλογή αυτή προέκυψε κυρίως από πρακτικούς λόγους: τα στοιχεία πρόσβασης της υπηρεσίας και η λογική που διαμορφώνει το αίτημα δεν είναι σωστό να βρίσκονται στον browser του χρήστη. Το Angular στέλνει στο API την εικόνα της λήψης και το backend αναλαμβάνει να τη συμπίεσει, να ετοιμάσει το αίτημα και να παραλάβει το αποτέλεσμα της ανάλυσης. [28]

Η εικόνα από μόνη της δεν είναι αρκετή για να τοποθετηθούν σωστά οι παρατηρήσεις πάνω στη διεπαφή. Τη στιγμή της λήψης, το YOLO έχει ήδη εντοπίσει ορισμένα ορατά στοιχεία, όπως κουμπιά, επικεφαλίδες, εικόνες, κάρτες ή τμήματα περιεχομένου. Για κάθε στοιχείο κρατιέται το είδος του, ένα αναγνωριστικό και η θέση του μέσα στην εικόνα. Η θέση περιγράφεται με bounding box, δηλαδή με τις συντεταγμένες ενός ορθογωνίου που περιβάλλει το στοιχείο. Αυτές οι πληροφορίες στέλνονται μαζί με την εικόνα στο backend μέσω του πεδίου uiElementsJson.

Το Gemini χρησιμοποιεί την εικόνα για να αξιολογήσει την οπτική ποιότητα της σελίδας, ενώ τα στοιχεία του YOLO λειτουργούν ως σημεία αναφοράς. Δεν ζητείται από το μοντέλο να σχεδιάσει από την αρχή νέα πλαίσια πάνω στην οθόνη. Ζητείται πρώτα να εξετάσει αν η παρατήρηση που θέλει να επιστρέψει σχετίζεται με κάποιο από τα στοιχεία που του έχουν δοθεί. Για παράδειγμα, ένα σχόλιο για μικρό κουμπί ή χαμηλή αντίθεση σε τίτλο μπορεί να συνδεθεί με το αντίστοιχο elementId. Στις περιπτώσεις όπου το πρόβλημα αφορά μια σειρά από παρόμοια στοιχεία, όπως αρκετές κάρτες προϊόντων, μπορεί να συνδεθεί με περισσότερα από ένα στοιχεία.

Η λογική αυτή ονομάζεται anchor-first. Στην πράξη, προηγείται η επιλογή ενός κατάλληλου anchor και ακολουθεί η δημιουργία του issue. Αν δεν υπάρχει αξιόπιστο στοιχείο με το οποίο να συνδεθεί η παρατήρηση, το issue επιστρέφεται μόνο ως κείμενο στην αναφορά. Δεν τοποθετείται αυθαίρετα πάνω στη ζωντανή εικόνα. Αυτό αποφεύγει περιπτώσεις στις οποίες ένα σωστό σχόλιο θα εμφανιζόταν σε λάθος σημείο της οθόνης και θα δημιουργούσε σύγχυση στον χρήστη.

Η ανάγκη για anchors συνδέεται άμεσα με τον τρόπο παρουσίασης των αποτελεσμάτων. Το ARTique δεν δείχνει μόνο μια λίστα προτάσεων. Όταν υπάρχει κατάλληλη αντιστοίχιση, προβάλλει πλαίσια πάνω στο στοιχείο που σχετίζεται με το πρόβλημα. Τα πλαίσια ακολουθούν την προοπτική της οθόνης που αναγνωρίζεται από το OpenCV.js, ώστε να παραμένουν προσαρμοσμένα στην εικόνα της κάμερας. Το YOLO δεν αποφασίζει αν η διεπαφή είναι σωστή ή λανθασμένη. Ο ρόλος του είναι χωρικός: εντοπίζει τις περιοχές στις οποίες μπορεί να τοποθετηθεί μια παρατήρηση. Η ερμηνεία του προβλήματος και η πρόταση βελτίωσης προέρχονται από το Gemini.

Η απάντηση του μοντέλου έχει συγκεκριμένη μορφή JSON. Περιλαμβάνει μια σύντομη περιγραφή του audit, τα issues που εντοπίστηκαν, τους περιορισμούς της ανάλυσης και το checklist της εφαρμογής. Κάθε issue περιέχει την κατηγορία του, τη σοβαρότητα, τον βαθμό εμπιστοσύνης, έναν τίτλο, περιγραφή του προβλήματος και πρόταση βελτίωσης. Όταν η σύνδεση με στοιχείο της διεπαφής είναι έγκυρη, επιστρέφονται επίσης το αναγνωριστικό του στοιχείου και το αντίστοιχο bounding box. Το checklist εξετάζει έντεκα κατηγορίες: αντίθεση, οπτική ιεραρχία, αποστάσεις, στοίχιση, μέγεθος περιοχών αφής, σαφήνεια call-to-action, αναγνωσιμότητα, οπτική συμφόρηση, πλοήγηση, συνέπεια και ορατές ενδείξεις προσβασιμότητας.

Το backend δεν προωθεί αυτούσια την απάντηση του Gemini στο frontend. Αρχικά ελέγχει αν οι τιμές του JSON συμφωνούν με το schema της εφαρμογής. Ελέγχει, για παράδειγμα, αν η κατηγορία και η σοβαρότητα είναι επιτρεπτές τιμές και αν το checklist δεν έρχεται σε αντίφαση με τα issues που επιστράφηκαν. Ελέγχει επίσης τα anchors. Αν το μοντέλο έχει συνδέσει ένα issue με στοιχείο που δεν υπάρχει στη λίστα ή έχει επιστρέψει αδύναμη αντιστοίχιση, το backend συγκρίνει τη θέση του issue με τις διαθέσιμες περιοχές του YOLO. Αν υπάρχει επαρκής επικάλυψη ή σαφώς κοντινότερο στοιχείο, η σύνδεση διορθώνεται πριν αποθηκευτεί και εμφανιστεί το αποτέλεσμα.

Η αξιολόγηση έχει αναγκαστικά οπτικά όρια. Το ARTique βλέπει την εικόνα μιας ιστοσελίδας, δεν έχει πρόσβαση στον κώδικά της. Μπορεί να σχολιάσει περιπτώσεις που φαίνονται στην οθόνη, όπως χαμηλή αντίθεση, αδύναμη ιεραρχία τίτλων, πυκνή διάταξη, μικρά κουμπιά ή δυσανάγνωστο κείμενο. Δεν μπορεί όμως να γνωρίζει αν η σελίδα έχει σωστά ARIA attributes, αν η πλοήγηση με πληκτρολόγιο λειτουργεί, αν η σειρά των στοιχείων είναι κατάλληλη για screen reader ή αν πληρούνται όλα τα κριτήρια WCAG. Επομένως, το αποτέλεσμα πρέπει να αντιμετωπίζεται ως τεκμηριωμένη οπτική αξιολόγηση και όχι ως πλήρης έλεγχος προσβασιμότητας. [2], [4]

3.8 Αποθήκευση Δεδομένων

Για τα αποτελέσματα των αναλύσεων χρησιμοποιείται MySQL. Στη βάση δεν αποθηκεύεται όλη η κατάσταση της εφαρμογής, αλλά κυρίως ό,τι χρειάζεται για να μπορεί να ξαναφορτωθεί ένα audit: η σύνοψη, τα προβλήματα που βρέθηκαν, οι περιορισμοί της ανάλυσης και η αρχική JSON απάντηση του μοντέλου. Αυτό καλύπτει και την απλή χρήση της εφαρμογής, όπου ο χρήστης θέλει να δει ξανά ένα προηγούμενο αποτέλεσμα, αλλά και την ανάγκη της διπλωματικής να υπάρχουν διαθέσιμα δεδομένα για μεταγενέστερο έλεγχο.

Κάθε audit έχει δύο αναγνωριστικά. Το πρώτο είναι το εσωτερικό id της βάσης. Το δεύτερο είναι το public_id, το οποίο χρησιμοποιείται στα API responses και στα requests του frontend. Στην πράξη, όταν το Angular ζητάει ένα παλιότερο audit, δεν χρειάζεται να γνωρίζει το auto-increment id της MySQL. Χρησιμοποιεί το public_id, που είναι πιο κατάλληλο για ανταλλαγή δεδομένων ανάμεσα σε frontend και backend.

Ο πίνακας audits κρατάει την κεντρική εγγραφή κάθε ανάλυσης. Εκεί μπαίνουν στοιχεία όπως το source_type, οι διαστάσεις της εικόνας, η σύνοψη, το overall_score, οι περιορισμοί και το raw_ai_response. Το overall_score υπάρχει στο schema, αλλά στην τρέχουσα έκδοση μένει null, επειδή δεν έχει οριστεί ακόμα μεθοδολογία βαθμολόγησης. Το raw_ai_response κρατάει ολόκληρη την απάντηση του Gemini. Το κράτησα έτσι γιατί κάποια χρήσιμα πεδία, όπως το checklist και τα στοιχεία anchoring, δεν υπάρχουν όλα ως ξεχωριστές στήλες.

Τα προβλήματα αποθηκεύονται στον πίνακα `audit_issues`. Κάθε γραμμή αντιστοιχεί σε ένα `issue` ενός συγκεκριμένου `audit`. Εκεί αποθηκεύονται η κατηγορία, η σοβαρότητα, ο τίτλος, το στοιχείο που αφορά, το πρόβλημα, η πρόταση βελτίωσης, το `confidence` και το αν η παρατήρηση είναι περισσότερο `objective` ή `subjective`. Όταν υπάρχει θέση πάνω στην εικόνα, αποθηκεύονται και οι τιμές του `bounding box`. Έτσι ένα `audit` μπορεί να εμφανιστεί ξανά χωρίς να γίνει νέα κλήση στο Gemini.

Ο τρίτος πίνακας είναι ο `audit_feedback`. Δεν είναι απαραίτητος για την απλή προβολή των αποτελεσμάτων, αλλά συνδέεται με την αξιολόγηση του συστήματος. Μπορεί να χρησιμοποιηθεί για να κρατηθεί `feedback` πάνω σε ένα `issue`, για παράδειγμα αν ήταν χρήσιμο, αν ήταν ακριβές ή αν η σοβαρότητα που δόθηκε ήταν σωστή. Στην τρέχουσα εφαρμογή αυτό το κομμάτι υπάρχει κυρίως ως υποδομή για επόμενο στάδιο. Ο Πίνακας 3.2 συνοψίζει βασικούς πίνακες της βάσης δεδομένων.

Πίνακας 3.2: Βασικοί πίνακες της βάσης δεδομένων

Πίνακας	Ρόλος στην εφαρμογή
<code>audits</code>	Κρατάει μία εγγραφή ανά <code>audit</code> , μαζί με <code>public_id</code> , σύνοψη, περιορισμούς, διαστάσεις εικόνας και <code>raw_ai_response</code> .
<code>audit_issues</code>	Κρατάει τα <code>issues</code> κάθε <code>audit</code> , μαζί με <code>category</code> , <code>severity</code> , <code>confidence</code> , πρόταση βελτίωσης και <code>bounding box</code> όπου υπάρχει.
<code>audit_feedback</code>	Κρατάει <code>feedback</code> για <code>issues</code> , ώστε να μπορεί να χρησιμοποιηθεί αργότερα στην αξιολόγηση του συστήματος.

3.9 Βασικές Σχεδιαστικές Αποφάσεις και Περιορισμοί

Κατά την ανάλυση η κάμερα παραμένει ενεργή. Το Gemini εξετάζει το στιγμιότυπο που έχει ληφθεί, ενώ το frontend συνεχίζει να προβάλλει τη ζωντανή εικόνα. Όταν επιστραφούν τα αποτελέσματα, οι επισημάνσεις τοποθετούνται στην περιοχή της οθόνης με την οποία συνδέεται κάθε παρατήρηση.

Η γεωμετρία αυτής της προβολής ξεκινά από το screen quad. Το OpenCV.js υπολογίζει τις τέσσερις γωνίες της οθόνης μέσα στην εικόνα της κάμερας. Με βάση αυτές παράγεται το crop στο οποίο εκτελείται το YOLO. Η ανίχνευση αφορά στοιχεία της διεπαφής, όπως κουμπιά, εικόνες, τίτλους και κάρτες. Η ίδια η οθόνη έχει ήδη εντοπιστεί σε προηγούμενο στάδιο.

Τη στιγμή της λήψης, το frontend στέλνει στο backend και τα στοιχεία που είχε εντοπίσει το YOLO, σε μορφή JSON. Κάθε στοιχείο συνοδεύεται από αναγνωριστικό και bounding box. Το Gemini χρησιμοποιεί αυτά τα δεδομένα για να συνδέσει ένα εύρημα με συγκεκριμένο τμήμα της διεπαφής. Η κρίση UI/UX προέρχεται από το Gemini· το YOLO παρέχει τη θέση. Πριν από την αποθήκευση, το backend ελέγχει αν το αναγνωριστικό και το bounding box του anchor συμφωνούν με τα στοιχεία που είχαν σταλεί από το frontend.

Η ακρίβεια της προβολής επηρεάζεται από το αρχικό screen quad. Μια μετατόπιση στις γωνίες του αλλάζει το crop και μεταφέρει το ίδιο σφάλμα στις θέσεις των overlays. Για αυτό χρησιμοποιείται σταθεροποίηση μεταξύ διαδοχικών frames. Υπάρχει επίσης έλεγχος της κίνησης της κάμερας. Όταν η λήψη θεωρείται ακίνητη, η γεωμετρία διατηρείται σταθερή αντί να ενημερώνεται από μικρές διαφορές κάθε νέου εντοπισμού. Ένας λανθασμένος αρχικός εντοπισμός παραμένει πιθανός, επομένως οι μηχανισμοί αυτοί αντιμετωπίζουν κυρίως την αστάθεια και όχι κάθε σφάλμα του detector.

Η εκτέλεση του YOLO μέσω ONNX Runtime Web επιβαρύνει περισσότερο έναν mobile browser από ό,τι έναν επιτραπέζιο υπολογιστή. Στο ίδιο περιβάλλον εκτελούνται η ροή της κάμερας, η απόδοση του canvas και η ενημέρωση των overlays. Για τον λόγο αυτό το μοντέλο δεν εκτελείται σε κάθε frame. Το tracking διατηρεί προσωρινά τις θέσεις που έχουν ήδη βρεθεί μέχρι να ολοκληρωθεί η επόμενη ανίχνευση. Η ταχύτητα και η σταθερότητα της ροής εξακολουθούν να επηρεάζονται από τη συσκευή και τον browser. [15]

Το υλικό που λαμβάνει το σύστημα είναι εικόνα. Δεν περιλαμβάνει το DOM, τους κανόνες CSS, τα ARIA attributes ή τη σειρά πλοήγησης με πληκτρολόγιο. Ούτε παρέχει πληροφορίες για τη συμπεριφορά της ιστοσελίδας σε screen reader. Η αξιολόγηση περιορίζεται σε όσα διακρίνονται οπτικά, όπως η αντίθεση, η αναγνωσιμότητα και το εμφανές μέγεθος ενός στόχου αφής. Τα ευρήματα αυτά δεν επαρκούν για έλεγχο πλήρους συμμόρφωσης με το WCAG. [2], [4]

Στο schema έχει προβλεφθεί το πεδίο overallScore, αλλά η τιμή του παραμένει null. Δεν έχει οριστεί τρόπος με τον οποίο διαφορετικές κατηγορίες και επίπεδα σοβαρότητας θα συνδυάζονται σε μία βαθμολογία. Μια σχετική μεθοδολογία θα χρειαζόταν σύγκριση με ανθρώπινη αξιολόγηση. Η τρέχουσα έκδοση αποθηκεύει κάθε εύρημα χωριστά, μαζί με τη σοβαρότητα, το confidence και την προτεινόμενη διόρθωση.

3.10 Συνοπτική αποτίμηση της αρχιτεκτονικής

Τα επιμέρους κομμάτια του ARtique συνδέονται σε μία διαδρομή. Η κάμερα και η προβολή διαχειρίζονται από το Angular. Το OpenCV.js βρίσκει την περιοχή της οθόνης μέσα στην εικόνα και το YOLO εντοπίζει εκεί τα ορατά στοιχεία της διεπαφής, δίνοντας τη θέση τους. Το Gemini εξετάζει το στιγμιότυπο και επιστρέφει τις παρατηρήσεις UI/UX. Το Node/Express back-end ελέγχει το JSON,

αντιστοιχίζει κάθε εύρημα με ένα από τα στοιχεία που στάλθηκαν από το frontend και αποθηκεύει το audit στη MySQL.

Η διάκριση ανάμεσα στην ανίχνευση και στην κριτική είναι σημαντική για το πώς ερμηνεύεται το πρωτότυπο. Το YOLO δίνει θέση και διαστάσεις ενός στοιχείου χωρίς να αξιολογεί την ποιότητά του. Μια παρατήρηση του Gemini μπορεί να είναι σωστή αλλά να μην αντιστοιχεί με ασφάλεια σε κάποιο anchor. Όταν συμβαίνει αυτό, το εύρημα μένει report-only και δεν εμφανίζεται ως πλαίσιο πάνω στην οθόνη.

3.11 Επίλογος

Η λειτουργία του ARtique χωρίζεται σε δύο ροές. Η πρώτη εκτελείται στον browser όσο η κάμερα είναι ανοιχτή. Σε αυτή τη ροή εντοπίζεται η οθόνη και πάνω της ανιχνεύονται τα στοιχεία της διεπαφής. Τα overlays σχεδιάζονται στο ίδιο camera view. Η δεύτερη ξεκινά με τη λήψη του στιγμιότυπου και περνά από το backend, το Gemini και τη MySQL.

Το ARtique αναπτύχθηκε ως ερευνητικό πρωτότυπο για την οπτική αξιολόγηση διεπαφών σε περιβάλλον WebAR. Στο επόμενο κεφάλαιο περιγράφεται η υλοποίηση αυτής της αρχιτεκτονικής και τα σημεία του κώδικα που αντιστοιχούν σε κάθε στάδιο.

Κεφάλαιο 4: Τεχνική Υλοποίηση του ARtique

4.1 Εισαγωγή

Το κεφάλαιο περιγράφει πώς υλοποιήθηκε το ARtique, ακολουθώντας την πορεία των δεδομένων μέσα στην εφαρμογή. Η περιγραφή ξεκινά από το στιγμιότυπο και τα στοιχεία που έχει εντοπίσει το YOLO, συνεχίζει με την ανάλυση και την αποθήκευση του audit και καταλήγει στην εμφάνιση των αποτελεσμάτων. Η λειτουργία του live WebAR εξετάζεται χωριστά, επειδή συνεχίζει να εκτελείται στο front-end ανεξάρτητα από το αίτημα προς το back-end.

Στην οθόνη σάρωσης η κάμερα δεν σταματά όσο αναμένεται το αποτέλεσμα. Το front-end εξακολουθεί να εντοπίζει την οθόνη, να σταθεροποιεί το screen quad, να εκτελεί το YOLO και να ενημερώνει τα overlays. Μόνο όταν πατηθεί το capture κρατιέται ένα στιγμιότυπο της ανιχνευμένης περιοχής. Μαζί του στέλνονται στο back-end τα YOLO elements που υπήρχαν εκείνη τη στιγμή, ώστε το Gemini να αναλύσει την ίδια εικόνα στην οποία θα αγκυρωθούν αργότερα τα issues.

Η είσοδος του συστήματος είναι η εικόνα της σελίδας και όχι ο κώδικάς της. Από αυτή την εικόνα μπορούν να εξεταστούν ορατά χαρακτηριστικά, όπως η αντίθεση, η ιεραρχία, η ευθυγράμμιση και η αναγνωσιμότητα. Δεν υπάρχουν όμως δεδομένα για τα ARIA attributes, την πλοήγηση με πληκτρολόγιο ή τη συμπεριφορά ενός screen reader. Για αυτό το αποτέλεσμα δεν παρουσιάζεται ως πλήρης έλεγχος WCAG, και ο περιορισμός αναφέρεται τόσο στο prompt όσο και στα limitations του audit.

4.2 Ανάπτυξη Back-End

Με το πάτημα του capture, το Angular δημιουργεί ένα multipart/form-data request που περιέχει την εικόνα και τα στοιχεία που εντόπισε το YOLO. Το request παραλαμβάνεται από το back-end, το οποίο έχει υλοποιηθεί με Node.js και Express. Εκεί η εικόνα συμπίεζεται και αποστέλλεται στο Gemini. Η απάντηση ελέγχεται ως προς τη δομή και τα anchors της πριν αποθηκευτεί το audit στη MySQL. Κατά την αναμονή, η κάμερα παραμένει ενεργή και τα AR overlays συνεχίζουν να υπολογίζονται στο front-end.

Η επεξεργασία ενός audit μοιράζεται σε διαφορετικά επίπεδα. Το route αντιστοιχίζει το endpoint με τον κατάλληλο controller, ο οποίος παραλαμβάνει τα δεδομένα και καλεί το αντίστοιχο service. Οι λειτουργίες που χρησιμοποιούνται σε περισσότερα σημεία, όπως το upload και η προεπεξεργασία της εικόνας, βρίσκονται στα helpers. Οι έλεγχοι του αποτελέσματος και η διόρθωση των anchors έχουν τοποθετηθεί στα utils, ενώ τα models αναλαμβάνουν την επικοινωνία με τη MySQL.

4.2.1 Οργάνωση Δομής Back-End

Το back-end οργανώνεται γύρω από το Express application που αρχικοποιείται στο server.js. Εκεί φορτώνονται τα βασικά middlewares, συνδέονται τα routes και ανοίγει η σύνδεση με τη MySQL. Από αυτό το σημείο και μετά, κάθε request περνά στο αντίστοιχο route και καταλήγει στον controller του φακέλου api. Ο controller δεν εκτελεί την ίδια την ανάλυση. Παραλαμβάνει τα δεδομένα που στέλνει το Angular, κάνει τον απαραίτητο χειρισμό του request και καλεί το AuditService, όπου βρίσκεται η κύρια ροή του audit.

Στο αίτημα δημιουργίας audit, το Upload middleware εκτελείται πριν από τον controller. Η υλοποίηση χρησιμοποιεί multer.memoryStorage(), επομένως η εικόνα παραδίδεται στο back-end ως buffer και δεν αποθηκεύεται πρώτα ως προσωρινό αρχείο στον δίσκο. Στη συνέχεια, το AuditService περνά το buffer

από το ImagePreprocessor, ώστε η εικόνα να συμπιεστεί και να προσαρμοστεί πριν σταλεί στο AIService. Το AIService χρησιμοποιεί το AuditPrompts για να δημιουργήσει το prompt με βάση τις διαστάσεις της εικόνας και τα YOLO elements που έχουν σταλεί μαζί με το request.

Όταν επιστρέψει η απάντηση από το μοντέλο, το back-end δεν τη χρησιμοποιεί απευθείας όπως είναι. Πρώτα κανονικοποιεί το checklist και μετά ελέγχει κάθε issue ξεχωριστά σε σχέση με τα YOLO elements του request. Αν το elementId του issue αντιστοιχεί σε υπαρκτό στοιχείο της λίστας, το σύστημα χρησιμοποιεί το bounding box αυτού του στοιχείου για την προβολή του marker. Αν δεν υπάρχει έγκυρη αντιστοιχία, το issue παραμένει μόνο στην αναφορά και δεν εμφανίζεται ως AR marker. Μετά τον έλεγχο αυτό, το audit και τα issues αποθηκεύονται στη MySQL μέσω των αντίστοιχων models. Στο Σχήμα 4.1 παρουσιάζεται δήλωση των endpoints των audits και χρήση του middleware μεταφόρτωσης εικόνας.

```

backend/src/api/routes/AuditRoutes.js
Verified repository excerpt

10 |   setRoutes() {
11 |     this.router.post('/audits/analyze', Upload.single('image'), AuditController.create);
12 |     this.router.get('/audits/:auditId/diagnostics', AuditController.diagnostics);
13 |     this.router.get('/audits/:auditId', AuditController.showOne);
14 |     this.router.get('/audits', AuditController.index);
15 |     this.router.delete('/audits/:auditId', AuditController.delete);
16 |   }
17 | }
18 |
19 | module.exports = AuditRoutes;

```

Σχήμα 4.1: Δήλωση των endpoints των audits και χρήση του middleware μεταφόρτωσης εικόνας.

Πηγή: Προσωπική υλοποίηση.

4.2.2 API Endpoints και Ροή Ανάλυσης

Η δημιουργία ενός audit ξεκινά με αίτημα POST /api/audits/analyze. Πριν φτάσει στον AuditController, το request περνά από το Upload.single('image'), το οποίο παραλαμβάνει το αρχείο της εικόνας. Η αποστολή γίνεται με multipart/form-data, ώστε το ίδιο request να περιλαμβάνει τόσο το δυαδικό αρχείο της εικόνας όσο και τα πεδία width, height, uiElementsJson και yoloConfigJson.

Στο uiElementsJson αποθηκεύεται η κατάσταση των YOLO detections τη στιγμή του capture. Κάθε έγκυρη εγγραφή περιλαμβάνει αναγνωριστικό, τύπο, ετικέτα, confidence και κανονικοποιημένο bounding box. Η λίστα αυτή χρησιμοποιείται στο anchor-first prompt ως το σύνολο των διαθέσιμων στοιχείων πάνω στα οποία μπορεί να αγκυρωθεί ένα issue. Έτσι, όταν η απάντηση του μοντέλου περιέχει elementId που υπάρχει στη λίστα, το back-end μπορεί να χρησιμοποιήσει το bounding box του αντίστοιχου στοιχείου. Όταν το αναγνωριστικό δεν είναι έγκυρο ή δεν υπάρχει επαρκής αντιστοιχία, το issue δεν προβάλλεται ως AR marker χωρίς νέο έλεγχο.

Τα υπόλοιπα endpoints του AuditRoutes χρησιμοποιούνται για την ανάκτηση ενός audit, την εμφάνιση του ιστορικού, τη διαγραφή μιας εγγραφής και την ανάγνωση των diagnostics. Το feedback αποθηκεύεται μέσω του POST /api/audits/:auditId/feedback, το οποίο έχει οριστεί ξεχωριστά στο FeedbackRoutes. Η σχετική φόρμα δεν έχει ακόμη προστεθεί στο front-end. Το endpoint μπορεί, ωστόσο, να καταγράψει αν ένα issue θεωρήθηκε χρήσιμο ή ακριβές, αν ο χρήστης συμφωνεί με τη σοβαρότητά του και τυχόν πρόσθετες παρατηρήσεις. Στο Σχήμα 4.2 παρουσιάζεται παραλαβή εικόνας και στοιχείων uiElementsJson από το request. Στο Σχήμα 4.3 παρουσιάζεται δημιουργία του multipart request από το Angular front-end.

backend/src/api/controllers/AuditController.js

Verified repository excerpt

```

10 |         throw new ValidationError('Image file is required');
11 |     }
12 |
13 |     const metadata = {
14 |       width: req.body.width ? parseInt(req.body.width) : null,
15 |       height: req.body.height ? parseInt(req.body.height) : null,
16 |       uiElements: parseClientUiElements(req.body.uiElementsJson),
17 |       yoloConfig: req.body.yoloConfigJson ? AuditController.parseJson(req.body.yoloConfigJson) : null,
18 |     };
19 |
20 |     const result = await AuditService.analyzeImage(req.file.buffer, metadata);

```

Σχήμα 4.2: Παραλαβή εικόνας και στοιχείων uiElementsJson από το request.

Πηγή: Προσωπική υλοποίηση.

WebAR Design Critic/src/app/services/audit.service.ts

Verified repository excerpt

```

20 |     height: number | null,
21 |     uiElements: DetectedUiElement[] = [],
22 | ): Observable<AuditResult> {
23 |   const blob = this.dataUrlToBlob(imageDataUrl);
24 |   const formData = new FormData();
25 |   formData.append('image', blob, 'capture.jpg');
26 |   if (width) formData.append('width', String(width));
27 |   if (height) formData.append('height', String(height));
28 |   if (uiElements.length > 0) {
29 |     formData.append('uiElementsJson', JSON.stringify(uiElements.map(element => ({
30 |       id: element.id,
31 |       trackId: element.trackId,
32 |       type: element.type,
33 |       label: element.label,
34 |       boundingBox: element.boundingBox,
35 |       confidence: element.confidence,
36 |       source: element.source,
37 |     })))));
38 |   }
39 |   formData.append('yoloConfigJson', JSON.stringify({
40 |     modelUrl: environment.yoloModelUrl,
41 |     inputSize: environment.yoloInputSize,
42 |     liveIntervalMs: environment.yoloLiveIntervalMs,
43 |     scoreThreshold: environment.yoloScoreThreshold,
44 |     iouThreshold: environment.yoloIouThreshold,

```

Σχήμα 4.3: Δημιουργία του multipart request από το Angular front-end.

Πηγή: Προσωπική υλοποίηση.

4.2.3 Επεξεργασία Εικόνας και Επικοινωνία με Gemini

Πριν από την κλήση του μοντέλου, το ImagePreprocessor επεξεργάζεται το buffer με τη βιβλιοθήκη sharp. Η μεγαλύτερη διάσταση περιορίζεται στα 1024 pixels και το αποτέλεσμα κωδικοποιείται ως

JPEG ποιότητας 82%. Αυτό το συμπιεσμένο buffer, και όχι το αρχικό αρχείο του request, περνά στην υπηρεσία AI.

Η μεταβλητή `AI_MODE` επιλέγει τη συμπεριφορά της υπηρεσίας AI. Στο `mock mode` επιστρέφεται σταθερό δοκιμαστικό `audit`, στο `sample mode` διαβάζεται αποθηκευμένη απάντηση και στο `real mode` εκτελείται κλήση προς το Gemini μέσω του native SDK. Τα δύο πρώτα modes χρησιμοποιήθηκαν για την ανάπτυξη της ροής στο Angular χωρίς να απαιτείται κλήση API σε κάθε δοκιμή.

Το prompt που στέλνεται στο Gemini είναι `anchor-first`. Αυτό σημαίνει ότι το μοντέλο δεν καλείται απλώς να γράψει γενικές παρατηρήσεις για την εικόνα, αλλά πρώτα να συσχετίσει κάθε πρόβλημα με κάποιο από τα UI στοιχεία που του έχουν δοθεί. Το αποτέλεσμα ζητείται σε αυστηρή JSON μορφή και περιλαμβάνει `summary`, λίστα `issues`, `checklist` κατηγοριών, `limitations` και στοιχεία αγκύρωσης όπως `elementId`, `anchorType`, `anchorConfidence` και `anchorReason`. Στο Σχήμα 4.4 παρουσιάζεται προεπεξεργασία και συμπίεση εικόνας πριν την αποστολή στο AI μοντέλο. Στο Σχήμα 4.5 παρουσιάζεται επιλογή λειτουργίας AI και κλήση της υπηρεσίας Gemini. Στο Σχήμα 4.6 παρουσιάζεται απόσπασμα του `anchor-first` prompt που δίνει στο Gemini τα `detected UI elements`.

```
backend/src/helpers/ImagePreprocessor.js
Verified repository excerpt

24 | static async compress(imageBuffer) {
25 |   const normalizedInput = sharp(imageBuffer).rotate();
26 |   const metadata = await normalizedInput.metadata();
27 |
28 |   const { width, height } = metadata;
29 |   const needsResize = width > MAX_DIMENSION || height > MAX_DIMENSION;
30 |
31 |   let pipeline = sharp(imageBuffer).rotate();
32 |
33 |   if (needsResize) {
34 |     pipeline = pipeline.resize(MAX_DIMENSION, MAX_DIMENSION, {
35 |       fit: 'inside',
36 |       withoutEnlargement: true,
37 |     });
38 |   }
39 |
40 |   const compressed = await pipeline
41 |     .jpeg({ quality: JPEG_QUALITY, mozjpeg: true })
42 |     .toBuffer();
43 | }
```

Σχήμα 4.4: Προεπεξεργασία και συμπίεση εικόνας πριν την αποστολή στο AI μοντέλο.

Πηγή: Προσωπική υλοποίηση.

backend/src/service/AIService.js

Verified repository excerpt

```

18
19 class AIService {
20   static async analyze(imageBuffer, options = {}) {
21     const mode = process.env.AI_MODE || 'mock';
22     Logger.info(`AIService.analyze - mode: ${mode}`);
23
24     switch (mode) {
25       case 'real':
26         return this.analyzeReal(imageBuffer, options);
27       case 'sample':
28         return this.loadSample(options);
29       case 'mock':
30
31     ...
32   static async analyzeWithGemini(imageBuffer, width, height, clientUiElements = [], options = {}) {
33     Logger.step('2/3 Building Gemini request (native SDK)...');
34     const apiKey = process.env.GEMINI_API_KEY;
35     if (!apiKey) throw new Error('Missing GEMINI_API_KEY environment variable');
36
37     const modelName = process.env.AI_MODEL || 'gemini-3.5-flash';
38     const fallbackName = process.env.AI_MODEL_FALLBACK || 'gemini-2.0-flash';
39     const modelCandidates = [modelName, ...(fallbackName && fallbackName !== modelName ? [fallbackName] : [])];
40     Logger.success(`Model: ${modelName}${modelCandidates.length > 1 ? ` (fallback: ${fallbackName})` : ''} : ...
41
42     const genAI = new GoogleGenerativeAI(apiKey);
43     // Primary model instance - used for UI-element grounding when needed.
44     const model = genAI.getGenerativeModel({
45       model: modelName,
46       systemInstruction: SYSTEM_PROMPT,

```

Σχήμα 4.5: Επιλογή λειτουργίας AI και κλήση της υπηρεσίας Gemini.

Πηγή: Προσωπική υλοποίηση.

```

backend/src/service/AuditPrompts.js
Verified repository excerpt

67 | const buildAuditPrompt = (imageWidth, imageHeight, uiElements = []) => `
68 | Analyze this web interface screenshot and return a structured UI/UX audit.
69 |
70 | This is prompt version ${PROMPT_VERSION}.
71 |
72 | ## Analysis checklist
73 | Systematically scan for issues in each category. Only report what is clearly visible:
74 |
75 | 1. contrast      - Is text/background contrast sufficient? Do interactive elements stand out?
76 | 2. hierarchy     - Is there a clear visual hierarchy across headings, subheadings, and body text?
77 | 3. spacing       - Is whitespace consistent? Do elements feel cramped or visually unbalanced?
78 | 4. alignment     - Are elements aligned to a consistent grid or shared edge?
79 | 5. touch-target  - Are interactive elements large enough for touch? (Minimum ~44x44 CSS px)
80 | 6. cta           - Is the primary call-to-action visually dominant and unambiguous?
81 | 7. readability  - Is line length, font size, and line-height comfortable for reading?
82 | 8. clutter       - Is the layout overcrowded? Does visual noise hurt focus?
83 | 9. navigation    - Is the navigation structure clear and scannable?
84 | 10. consistency - Do repeated elements (cards, buttons, icons) look visually consistent?
85 | 11. accessibility - Visible concerns: color-only information, missing focus rings, very small text?
86 |
87 | ## Bounding box format
88 | The image is ${imageWidth}px wide x ${imageHeight}px tall.
89 | Coordinates are normalized to 0.0-1.0 relative to image dimensions:
90 | - x, y          : top-left corner of the issue area ÷ image width/height
91 | - width, height : size of the area ÷ image width/height
92 | - Use null if the area cannot be reliably located.

```

Σχήμα 4.6: Απόσπασμα του anchor-first prompt που δίνει στο Gemini τα detected UI elements.

Πηγή: Προσωπική υλοποίηση.

4.2.4 Κανονικοποίηση, Anchoring και Αποθήκευση

Μετά την απάντηση του Gemini, το back-end δεν επιστρέφει άμεσα το αποτέλεσμα στο front-end. Πρώτα εκτελείται έλεγχος και κανονικοποίηση. Οι κατηγορίες των προβλημάτων περιορίζονται σε συγκεκριμένη λίστα, όπως contrast, hierarchy, spacing, alignment, touch-target, cta, readability, clutter, navigation, consistency και accessibility. Παράλληλα, το checklist επιστρέφεται πάντα με τις ίδιες έντεκα κατηγορίες, ώστε το UX Report panel να έχει σταθερή δομή.

Ιδιαίτερη σημασία έχει το grounding repair. Αν το Gemini επιστρέψει issue με elementId που δεν αντιστοιχεί σωστά στα στοιχεία που δόθηκαν, το back-end μπορεί να διορθώσει την αγκύρωση με βάση overlap ή κοντινότερο centroid. Αυτό κάνει την τοποθέτηση των AR boxes πιο αξιόπιστη και μειώνει την πιθανότητα ένα πρόβλημα να εμφανιστεί πάνω σε λάθος περιοχή της διεπαφής. Όταν δεν υπάρχει ασφαλές οπτικό σημείο, το issue μπορεί να χαρακτηριστεί ως report-only και να εμφανιστεί μόνο στην αναφορά.

Τα τελικά δεδομένα αποθηκεύονται στη MySQL. Ο πίνακας audits κρατά την κεντρική εγγραφή της ανάλυσης, ενώ ο πίνακας audit_issues κρατά τα επιμέρους προβλήματα. Το raw_ai_response διατηρεί πεδία που δεν έχουν ξεχωριστές στήλες, όπως το checklist και οι πληροφορίες anchoring. Η εικόνα υπάρχει μόνο στη μνήμη κατά την επεξεργασία του request και δεν καταχωρίζεται στη βάση στην τρέχουσα έκδοση. Στο Σχήμα 4.7 παρουσιάζεται διόρθωση και επαλήθευση anchoring των AI issues. Στο Σχήμα 4.8 παρουσιάζεται κανονικοποίηση του UX checklist με τις έντεκα σταθερές κατηγορίες. Στο Σχήμα 4.9 παρουσιάζεται δημιουργία των βασικών πινάκων MySQL για audits, audit_issues και audit_feedback.

backend/src/utils/groundingRepair.js

Verified repository excerpt

```

119 function repairIssueAnchors(rawIssues = [], uiElements = []) {
120   const validElements = Array.isArray(uiElements) ? uiElements.filter(element => normalizeBox(element.boundin...
121   const byId = new Map(validElements.map(element => [element.id, element]));
122   const diagnostics = [];
123
124   const issues = (Array.isArray(rawIssues) ? rawIssues : []).map((issue) => {
125     const base = normalizeAnchorFields(issue);
126     const requestedElement = issue.elementId ? byId.get(issue.elementId) : null;
127     const inputBox = normalizeBox(issue.boundingBox);
128
129     if (requestedElement) {
130       const repaired = {
131         ...issue,
132         elementId: requestedElement.id,
133         affectedElementIds: [requestedElement.id, ...base.affectedElementIds.filter(id => byId.has(id) && id ...
134         anchorType: 'single',
135         anchorSource: 'gemini',
136         anchorConfidence: requestedElement.confidence || base.anchorConfidence || 'medium',
137         anchorReason: base.anchorReason || `Gemini selected detected element ${requestedElement.id}.`,
138         reportOnly: false,
139         boundingBox: { ...requestedElement.boundingBox },
140       };
141       diagnostics.push(makeDiagnostic(issue, repaired, 'valid-gemini-element-id'));
142       return repaired;
143     }
144
145     if (inputBox && validElements.length > 0) {
146       const group = groupElementsForBox(inputBox, validElements);
147       if (group.length >= 2 && boxArea(inputBox) > Math.max(...group.map(element => boxArea(element.boundingB...

```

Σχήμα 4.7: Διόρθωση και επαλήθευση anchoring των AI issues.

Πηγή: Προσωπική υλοποίηση.

backend/src/utils/checklistCategories.js

Verified repository excerpt

```

18 function normalizeChecklist(rawChecklist, issues) {
19   const byCat = new Map();
20   if (Array.isArray(rawChecklist)) {
21     for (const row of rawChecklist) {
22       if (row && typeof row.category === 'string') byCat.set(row.category, row);
23     }
24   }
25   const withIssues = new Set((issues || []).map(i => i && i.category).filter(Boolean));
26
27   return CANONICAL_CATEGORIES.map(category => {
28     const ai = byCat.get(category);
29     const hasIssue = withIssues.has(category);
30     return {
31       category,
32       status: hasIssue ? 'issue' : (ai && ai.status === 'issue' ? 'issue' : 'good'),
33       note: (ai && typeof ai.note === 'string') ? ai.note : '',
34     };
35   });
36 }
37

```

Σχήμα 4.8: Κανονικοποίηση του UX checklist με τις έντεκα σταθερές κατηγορίες.

Πηγή: Προσωπική υλοποίηση.

backend/src/database/tables.js

Verified repository excerpt

```

1  const tables = [
2    {
3      name: 'audits',
4      sql: `CREATE TABLE IF NOT EXISTS \`${audits}\` (
5        \`${id}\` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
6        \`${public_id}\` VARCHAR(64) NOT NULL,
7        \`${source_type}\` VARCHAR(32) NOT NULL DEFAULT 'camera_capture',
8        \`${image_path}\` VARCHAR(500) NULL,
9        \`${image_width}\` INT UNSIGNED NULL,
10       \`${image_height}\` INT UNSIGNED NULL,
11       \`${summary}\` TEXT NULL,
12       \`${overall_score}\` DECIMAL(5,2) NULL,
13       \`${limitations}\` JSON NULL,
14       \`${raw_ai_response}\` JSON NULL,
15       \`${created_at}\` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
16       \`${updated_at}\` TIMESTAMP NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
17     ...
25     sql: `CREATE TABLE IF NOT EXISTS \`${audit_issues}\` (
26       \`${id}\` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
27       \`${audit_id}\` BIGINT UNSIGNED NOT NULL,
28       \`${issue_key}\` VARCHAR(64) NOT NULL,
29       \`${category}\` VARCHAR(64) NOT NULL,
30       \`${severity}\` ENUM('low', 'medium', 'high') NOT NULL,
31       \`${title}\` VARCHAR(255) NOT NULL,
32       \`${element}\` VARCHAR(255) NULL,
33       \`${bbox_x}\` DECIMAL(8,6) NULL,
34       \`${bbox_y}\` DECIMAL(8,6) NULL,
35       \`${bbox_width}\` DECIMAL(8,6) NULL,
36       \`${bbox_height}\` DECIMAL(8,6) NULL,
37     ...
50     sql: `CREATE TABLE IF NOT EXISTS \`${audit_feedback}\` (
51       \`${id}\` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
52       \`${audit_id}\` BIGINT UNSIGNED NOT NULL,
53       \`${issue_id}\` BIGINT UNSIGNED NULL,
54       \`${useful}\` TINYINT(1) NULL,
55       \`${accurate}\` TINYINT(1) NULL,
56       \`${severity_agreement}\` ENUM('too_low', 'correct', 'too_high') NULL,
57       \`${notes}\` TEXT NULL,
58       \`${created_at}\` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
59       PRIMARY KEY (\`${id}\`),
60       INDEX \`${idx_feedback_audit}\` (\`${audit_id}\`),

```

Σχήμα 4.9: Δημιουργία των βασικών πινάκων MySQL για audits, audit_issues και audit_feedback.

Πηγή: Προσωπική υλοποίηση.

4.3 Ανάπτυξη Front-End

Το front-end αναπτύχθηκε με Angular και εκτελεί τη ζωντανή ροή της εφαρμογής. Διαχειρίζεται το camera stream, την ανίχνευση οθόνης, το live YOLO inference, τα overlays και τις οθόνες αναφοράς. Η οθόνη scan έχει σχεδιαστεί κυρίως για κινητό, επειδή το σενάριο χρήσης απαιτεί η κάμερα της συσκευής να στραφεί προς ιστοσελίδα που προβάλλεται σε δεύτερη οθόνη. [20]

Η δομή του Angular project βασίζεται σε NgModules και lazy-loaded feature modules. Υπάρχει αρχική οθόνη, οθόνη σάρωσης, οθόνη αποτελεσμάτων audit και οθόνη ιστορικού. Τα services είναι υπεύθυνα για λειτουργίες όπως η κάμερα, η λήψη εικόνας, η ανίχνευση οθόνης, το live YOLO inference, η επικοινωνία με το back-end και η διατήρηση της κατάστασης του audit μεταξύ των οθονών.

4.3.1 Αρχιτεκτονική και Οντότητες Front-End

Το `app-routing.module.ts` περιέχει τις τέσσερις κύριες διαδρομές του front-end. Η αρχική σελίδα δηλώνεται απευθείας στο `AppModule`. Οι διαδρομές `/scan`, `/audit/:id` και `/audits` φορτώνουν τα δικά τους feature modules μόνο όταν ο χρήστης τις ανοίξει. Η πρώτη χρησιμοποιείται για τη σάρωση, η δεύτερη για την εμφάνιση ενός audit μέσω του public id του και η τρίτη για το ιστορικό των αποθηκευμένων αναλύσεων.

Τα `AuditResult` και `AuditIssue` περιγράφουν τη μορφή των δεδομένων που λαμβάνει το Angular από το API. Για τη λειτουργία της κάμερας και του live pipeline χρησιμοποιούνται ξεχωριστά models για τα YOLO detections, τα bounding boxes, τα anchored issues και το camera state. Οι πληροφορίες αυτές υπάρχουν μόνο όσο εκτελείται η σάρωση και δεν αποτελούν μέρος του αποθηκευμένου audit.

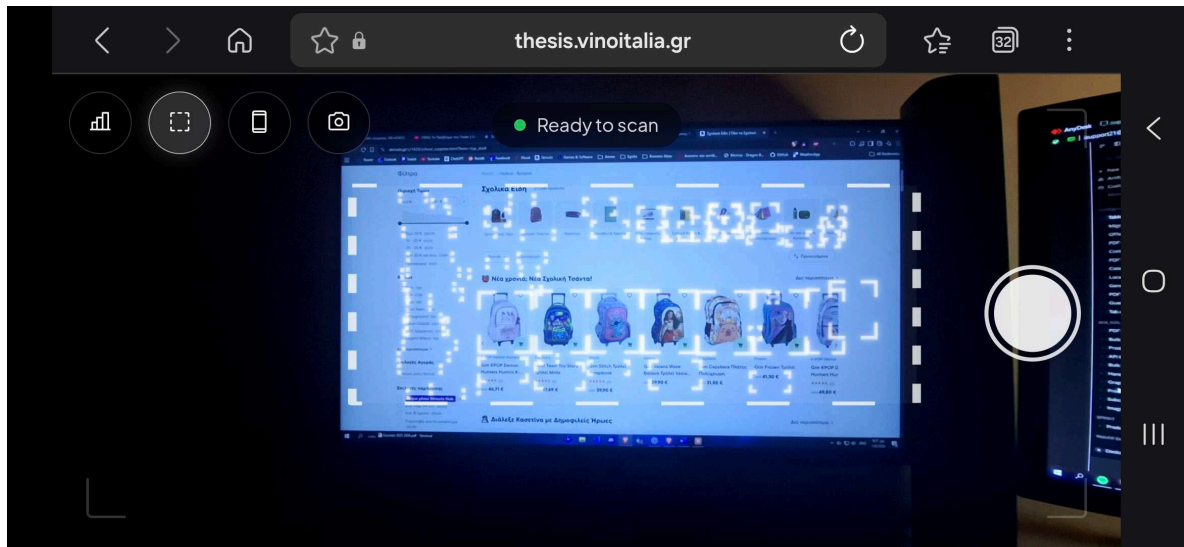
Το scan component διατηρεί την κατάσταση της σάρωσης και μοιράζει τα δεδομένα στα επιμέρους components. Το camera-preview εμφανίζει το video stream και το detection-overlay σχεδιάζει τις detections που παράγει το YOLO. Όταν ολοκληρωθεί η ανάλυση, το ar-marker-layer προβάλλει τα issues που διαθέτουν anchor, ενώ το ux-report-panel εμφανίζει το checklist και τις περιγραφές τους. Για την προβολή ενός αποθηκευμένου αποτελέσματος χρησιμοποιείται το audit-results, το οποίο ζητά από το API το audit και τα issues που ανήκουν σε αυτό.

4.3.2 Ροή Σάρωσης και Διαχείριση Κάμερας

Με την είσοδο στη διαδρομή `/scan`, το camera service ζητά από τον browser πρόσβαση στην κάμερα της συσκευής. Αν το αίτημα γίνει δεκτό, το video stream συνδέεται με το camera-preview. Η άρνηση της άδειας και η αδυναμία εύρεσης κάμερας εμφανίζονται χωριστά, καθώς απαιτούν διαφορετική ενέργεια από τον χρήστη. [22]

Η κάμερα δεν σταματά όταν πατηθεί το capture. Εκείνη τη στιγμή το scan component κρατά την εικόνα της περιοχής που έχει εντοπιστεί ως οθόνη και τα ενεργά YOLO elements. Τα δεδομένα αποστέλλονται για ανάλυση, ενώ το live pipeline εξακολουθεί να επεξεργάζεται το camera feed. Με την επιστροφή του αποτελέσματος, τα έγκυρα anchors μετατρέπονται σε θέσεις επάνω στο πιο πρόσφατο σταθεροποιημένο screen quad.

Στην οθόνη σάρωσης υπάρχει επίσης ένδειξη ετοιμότητας, μήνυμα για landscape προσανατολισμό και επιλογή ενεργοποίησης του torch στις συσκευές που το υποστηρίζουν. Η ετοιμότητα δεν αποτελεί προϋπόθεση για το capture. Χρησιμοποιείται μόνο ως ένδειξη ότι το live pipeline έχει ήδη εντοπίσει στοιχεία τα οποία μπορούν να αξιοποιηθούν ως anchors στο αποτέλεσμα. Στο Σχήμα 4.10 παρουσιάζεται κατάσταση Ready to scan με κλειδωμένο screen quad και ορατά στοιχεία διεπαφής σε πραγματική κινητή συσκευή.



Σχήμα 4.10: Κατάσταση Ready to scan με κλειδωμένο screen quad και ορατά στοιχεία διεπαφής σε πραγματική κινητή συσκευή.

Πηγή: Προσωπική υλοποίηση.

4.3.3 Live WebAR Pipeline και Ανίχνευση Οθόνης

Το screen-detector service λαμβάνει διαδοχικά frames από το camera feed και τα επεξεργάζεται με OpenCV.js. Για τον εντοπισμό της οθόνης εξετάζει τις ακμές, τη φωτεινή περιοχή της εικόνας και τη γεωμετρία των διαθέσιμων περιγραμμάτων. Από τους υποψήφιους συνδυασμούς επιλέγει εκείνον με την καλύτερη βαθμολογία και επιστρέφει τις τέσσερις γωνίες του. Σε αυτή τη διαδικασία δεν εξετάζεται το περιεχόμενο της ιστοσελίδας. Παράγεται μόνο το screen quad, το οποίο ορίζει την περιοχή εισόδου του YOLO και τις συντεταγμένες επάνω στις οποίες προβάλλονται τα overlays.

Πριν χρησιμοποιηθεί, κάθε νέο quad περνά από τον screen-quad stabilizer. Μικρές μετακινήσεις των γωνιών που βρίσκονται μέσα στο corner deadband αγνοούνται, ώστε να διατηρείται η προηγούμενη θέση. Αν η διαφορά είναι μεγαλύτερη, το νέο quad πρέπει να εμφανιστεί ξανά πριν γίνει αποδεκτό. Αντίστοιχα, η ευθυγράμμιση των αξόνων ενεργοποιείται και απενεργοποιείται με διαφορετικά όρια μέσω hysteresis. Όταν το camera-motion gate διαπιστώσει ότι η συσκευή έχει παραμείνει ακίνητη, η θέση παγώνει μέχρι να χρειαστεί relock.

Χωρίς αυτή τη σταθεροποίηση, ακόμη και μικρές διαφορές ανάμεσα σε διαδοχικά frames θα μετακινούσαν συνεχώς το περίγραμμα και τα AR markers. Το αποτέλεσμα θα έμοιαζε με τρέμουλο ή ολίσθηση των επισημάνσεων επάνω στη διεπαφή. Για τον λόγο αυτό, το περίγραμμα της οθόνης και τα markers προβάλλονται με βάση το ίδιο σταθεροποιημένο quad και ακολουθούν μαζί την κίνηση της κάμερας. Στο Σχήμα 4.11 παρουσιάζεται ανίχνευση οθόνης στο live camera feed. Στο Σχήμα 4.12 παρουσιάζεται μηχανισμός σταθεροποίησης του screen quad με deadband, freeze και relock. Στο Σχήμα 4.13 παρουσιάζεται έλεγχος κίνησης κάμερας που επιτρέπει το πάγωμα του quad όταν η συσκευή είναι ακίνητη.

WebAR Design Critic/src/app/services/screen-detector.service.ts

Verified repository excerpt

```

278 | // Measure processing time
279 | const t0 = performance.now();
280 |
281 | // Draw the current video frame once; both detection and tracking read it.
282 | const framed = this.captureFrame();
283 | const cameraStill = framed ? this.updateCameraMotion() : false;
284 |
285 | let result: Point[] | null = null;
286 | let trackedThisFrame = false;
287 |
288 | ...
330 | // Blend same-region candidates, hold against transient outliers, snap on a
331 | // persistent region change – never average two different physical quads.
332 | const stabRes = updateQuadStabilizer(this.stabilizer, result, this.stabilizerConfig, cameraStill);
333 | this.stabilizer = stabRes.state;
334 | this.lastStabilizerAction = stabRes.action;
335 |
336 | const locked = this.consecutiveDetections >= this.LOCK_THRESHOLD;
337 | // A straight-held phone on a straight monitor must produce a straight quad:

```

Σχήμα 4.11: Ανίχνευση οθόνης στο live camera feed.

Πηγή: Προσωπική υλοποίηση.

WebAR Design Critic/src/app/core/utils/screen-quad-stabilizer.util.ts

Verified repository excerpt

```

309 | export function updateQuadStabilizer(
310 |   state: QuadStabilizerState,
311 |   candidate: Point[] | null,
312 |   config: QuadStabilizerConfig = DEFAULT_QUAD_STABILIZER_CONFIG,
313 |   cameraStill = false,
314 | ): QuadStabilizerResult {
315 |   const current: QuadStabilizerState = {
316 |     quad: state.quad ? copyQuad(state.quad) : null,
317 |     pendingQuad: state.pendingQuad ? copyQuad(state.pendingQuad) : null,
318 |     pendingCount: state.pendingCount,
319 |     pendingBlendQuad: state.pendingBlendQuad ? copyQuad(state.pendingBlendQuad) : null,
320 |     pendingBlendCount: state.pendingBlendCount,
321 |   };
322 |
323 |   if (!candidate || candidate.length !== 4) {
324 |     return { state: current, action: 'none' };
325 |   }
326 |
327 |   if (!current.quad) {
328 |     return {
329 |       state: { ...createQuadStabilizerState(), quad: copyQuad(candidate) },
330 |       action: 'init',
331 |     };
332 |   }
333 |
334 |   const iou = quadIoU(current.quad, candidate);
335 |
336 |   if (iou >= config.blendIoU) {
337 |     const delta = maxCornerDelta(current.quad, candidate);
338 |
339 |     // Camera confirmed still: the true screen cannot move, so every same-region
340 |     // delta is noise (contour jitter, optical-flow wander in a lit room). Freeze
341 |     // the quad exactly – never blend toward the noise. Only a large, persistent
342 |     // divergence (the relock path below) may move it.
343 |     if (cameraStill) {

```

Σχήμα 4.12: Μηχανισμός σταθεροποίησης του screen quad με deadband, freeze και relock.

Πηγή: Προσωπική υλοποίηση.

```

WebAR Design Critic/src/app/core/utils/camera-motion.util.ts
Verified repository excerpt

142 | export function updateMotionGate(
143 |   state: CameraMotionState,
144 |   motion: number,
145 |   config: CameraMotionConfig = DEFAULT_CAMERA_MOTION_CONFIG,
146 | ): MotionGateResult {
147 |   if (state.cameraStill) {
148 |     if (motion > config.movingExitThreshold) {
149 |       return { state: { stillFrames: 0, cameraStill: false }, cameraStill: false };
150 |     }
151 |     return { state: { stillFrames: state.stillFrames + 1, cameraStill: true }, cameraStill: true };
152 |   }
153 |
154 |   if (motion < config.stillEnterThreshold) {
155 |     const stillFrames = state.stillFrames + 1;
156 |     const cameraStill = stillFrames >= config.stillMinFrames;
157 |     return { state: { stillFrames, cameraStill }, cameraStill };
158 |   }
159 |   return { state: { stillFrames: 0, cameraStill: false }, cameraStill: false };
160 | }

```

Σχήμα 4.13: Έλεγχος κίνησης κάμερας που επιτρέπει το πάγωμα του quad όταν η συσκευή είναι ακίνητη.

Πηγή: Προσωπική υλοποίηση.

4.3.4 Ανίχνευση Στοιχείων UI με YOLO

Το YOLO δεν επεξεργάζεται ολόκληρο το camera frame. Οι τέσσερις γωνίες του σταθεροποιημένου screen quad χρησιμοποιούνται για να παραχθεί ένα ορθογώνιο crop της οθόνης, και αυτό είναι το frame που δίνεται στο μοντέλο. Το αρχείο του μοντέλου είναι σε μορφή ONNX και φορτώνεται από το ONNX Runtime Web. Όπου υποστηρίζεται OffscreenCanvas, το inference μεταφέρεται σε Web Worker ώστε να μην εκτελείται στο κύριο thread της Angular εφαρμογής. [15]

Κάθε detection του YOLO περιέχει την κατηγορία του στοιχείου, το confidence και το bounding box του. Το tracking διατηρεί ένα αναγνωριστικό για όσα στοιχεία παραμένουν ορατά και αυτά τα αναγνωριστικά στέλνονται μαζί με την εικόνα. Η λίστα χρησιμοποιείται από το Gemini ως σύνολο διαθέσιμων anchors, ώστε ένα issue να μπορεί να συνδεθεί με συγκεκριμένο elementId. Το YOLO δίνει τη θέση του στοιχείου· η ερμηνεία του προβλήματος και η πρόταση διόρθωσης ανήκουν στο Gemini.

Για να μη δημιουργείται νέο box από κάθε στιγμιαία detection, ένα track εμφανίζεται αφού επιβεβαιωθεί σε δύο frames. Το όριο εισόδου είναι 0,25, ενώ ένα υπάρχον track μπορεί να διατηρηθεί με score από 0,12 και πάνω. Αν χαθεί προσωρινά, παραμένει διαθέσιμο για έως τρία δευτερόλεπτα. Παράλληλα εφαρμόζεται εξομάλυνση του bounding box και τα θολά frames παραλείπονται χωρίς να θεωρούνται χαμένα τα ενεργά tracks. Οι ρυθμίσεις αυτές περιορίζουν το τρέμουλο, δεν διορθώνουν όμως τα λάθη του ίδιου του μοντέλου. Η πρώτη εκπαίδευση έγινε με μικρό dataset, γι' αυτό δημιουργήθηκε δεύτερη διαδικασία παραγωγής δεδομένων με παραμόρφωση προοπτικής, θόρυβο, glare και moire. Στο Σχήμα 4.14 παρουσιάζεται εκτέλεση YOLO στο browser μέσω ONNX Runtime Web. Στο Σχήμα 4.15 παρουσιάζεται tracking και εξομάλυνση των YOLO bounding boxes. Στο Σχήμα 4.16 παρουσιάζεται παραγωγή camera-augmented dataset για καλύτερη προσαρμογή σε λήψεις οθόνης από κινητό.

WebAR Design Critic/src/app/services/live-yolo-inference.worker.ts

Verified repository excerpt

```

83 | function getSession(modelUrl: string, wasmPaths: string): Promise<ort.InferenceSession> {
84 |   if (!sessionPromise || configuredModelUrl !== modelUrl) {
85 |     ort.env.wasm.wasmPaths = wasmPaths;
86 |     configuredModelUrl = modelUrl;
87 |     sessionPromise = ort.InferenceSession.create(modelUrl, {
88 |       executionProviders: ['wasm'],
89 |       graphOptimizationLevel: 'all',
90 |     });
91 |   }
92 |   return sessionPromise;

```

Σχήμα 4.14: Εκτέλεση YOLO στο browser μέσω ONNX Runtime Web.

Πηγή: Προσωπική υλοποίηση.

WebAR Design Critic/src/app/core/utlils/live-yolo.util.ts

Verified repository excerpt

```

191 | export function updateYoloTracks(
192 |   previous: DetectedUiElement[],
193 |   detections: DetectedUiElement[],
194 |   options: YoloTrackUpdateOptions = {},
195 | ): YoloTrackUpdateResult {
196 |   const config = { ...DEFAULT_YOLO_TRACKING_CONFIG, ...options.config };
197 |   const now = options.now ?? performance.now();
198 |   const lowConfidenceDetections = options.lowConfidenceDetections ?? [];
199 |
200 |   const tracks: DetectedUiElement[] = [];
201 |   const remainingTracks = new Set(previous);
202 |
203 |   const firstPass = associateDetections(remainingTracks, detections, config);
204 |   for (const [track, detection] of firstPass.matches) {
205 |     tracks.push(mergeDetectionIntoTrack(track, detection, now, config));
206 |   }
207 |
208 |   const secondPass = associateDetections(remainingTracks, lowConfidenceDetections, config);
209 |   for (const [track, detection] of secondPass.matches) {
210 |     tracks.push(mergeDetectionIntoTrack(track, detection, now, config));
211 |   }
212 |
213 |   for (const detection of firstPass.unmatchedDetections) {
214 |     trackCounter += 1;
215 |     tracks.push({
216 |       ...detection,
217 |       id: `el-live-${trackCounter}`,
218 |       trackId: `yolo-track-${trackCounter}`,
219 |       rawConfidence: detection.confidence,
220 |       hitCount: 1,
221 |       lastSeenAt: now,
222 |     });
223 |   }
224 |
225 |   for (const track of remainingTracks) {
226 |     if ((track.hitCount ?? 0) < config.minHitsToDisplay) continue;

```

Σχήμα 4.15: Tracking και εξομάλυνση των YOLO bounding boxes.

Πηγή: Προσωπική υλοποίηση.

ml/ui-yolo/ui_yolo/camera_augmentation.py

Verified repository excerpt

```

114 def augment_dataset_for_camera(
115     input_dir: Path,
116     output_dir: Path,
117     variants_per_image: int = 2,
118     max_images: int = 0,
119     overwrite: bool = False,
120     augment_val: bool = False,
121     seed: int = 42,
122 ) -> dict[str, int]:
123     rng = random.Random(seed)
124     input_dir = Path(input_dir)
125     output_dir = Path(output_dir)
126     if not (input_dir / "images" / "train").exists():
127         raise FileNotFoundError(f"Expected YOLO dataset at {input_dir}")
128
129     prepare_yolo_folders(output_dir, overwrite=overwrite)
130     class_names = parse_data_yaml_names(input_dir / "data.yaml")
131     if not class_names:
132         class_names = ["button", "text_field", "image", "checkbox"]
133
134     manifest: dict[str, object] = {
135         "source_dataset": str(input_dir.resolve()),
136         "variants_per_image": variants_per_image,
137         "augment_val": augment_val,
138         "items": [],
139     }
140     counts = {
141         "train_original": 0,
142         "train_augmented": 0,
143         "val_original": 0,
144         "val_augmented": 0,
145     }

```

Σχήμα 4.16: Παραγωγή camera-augmented dataset για καλύτερη προσαρμογή σε λήψεις οθόνης από κινητό.

Πηγή: Προσωπική υλοποίηση.

4.3.5 Οθόνες Διεπαφής Χρήστη και Περιπτώσεις Χρήσης

Οι περιπτώσεις χρήσης που ακολουθούν αντιστοιχούν στις βασικές οθόνες και διαδρομές της εφαρμογής. Η διαδικασία αρχίζει από την αρχική σελίδα, περνά στη ζωντανή σάρωση και ολοκληρώνεται με την προβολή των αποτελεσμάτων. Ο χρήστης μπορεί επίσης να ανοίξει ξανά ένα audit από το ιστορικό, χωρίς να επαναλάβει την ανάλυση.

ΠΧ-1 (Έναρξη νέας σάρωσης): Από την αρχική οθόνη, ο χρήστης επιλέγει να ξεκινήσει μια νέα σάρωση και μεταφέρεται στη διαδρομή /scan.

ΠΧ-2 (Παραχώρηση άδειας κάμερας): Κατά την είσοδο στην οθόνη σάρωσης, ο browser ζητά πρόσβαση στην κάμερα. Αν δοθεί η άδεια, εμφανίζεται το ζωντανό video. Σε περίπτωση άρνησης ή απουσίας διαθέσιμης κάμερας, προβάλλεται το αντίστοιχο μήνυμα σφάλματος.

ΠΧ-3 (Ανίχνευση οθόνης και στοιχείων UI): Κατά την προβολή του camera feed, η εφαρμογή εντοπίζει και σταθεροποιεί το screen quad. Παράλληλα, το YOLO αναγνωρίζει τα ορατά στοιχεία της διεπαφής και καταγράφει τις θέσεις τους.

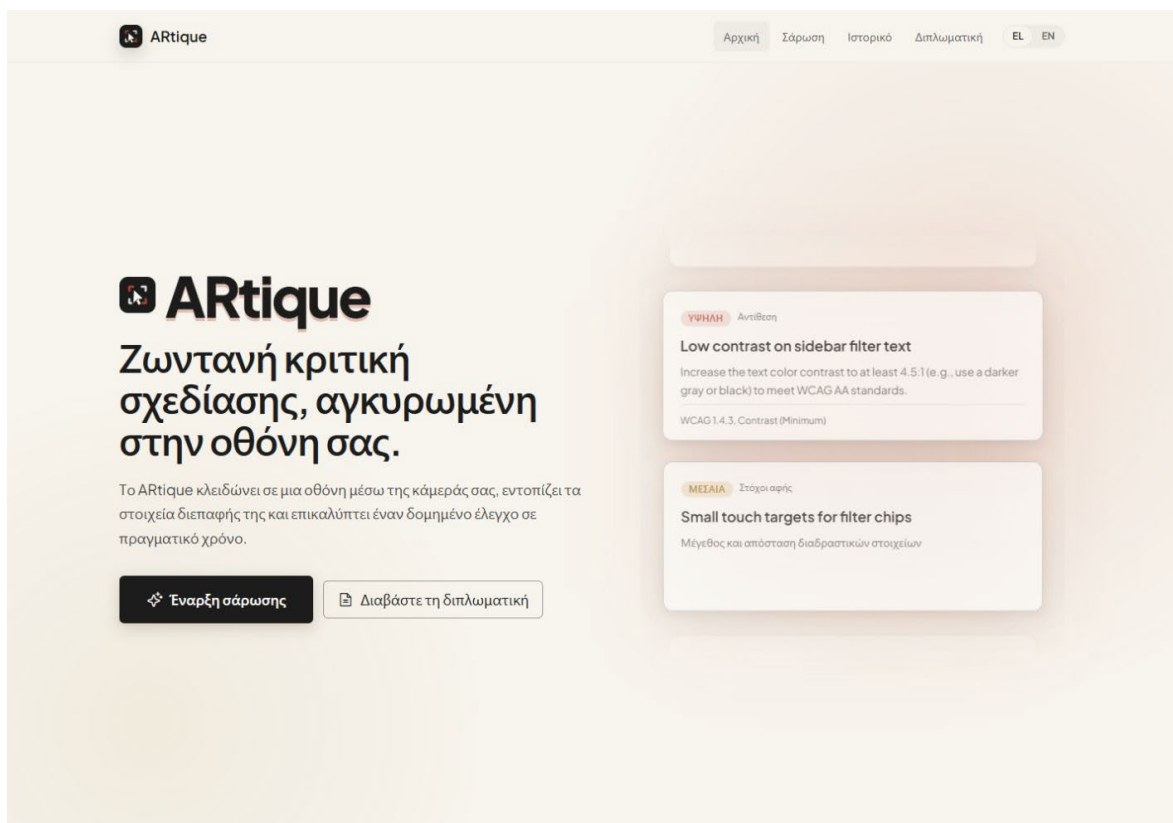
ΠΧ-4 (Υποβολή για αξιολόγηση): Με την επιλογή capture, δημιουργείται στιγμιότυπο της ανιχνευμένης οθόνης. Η εικόνα και τα διαθέσιμα YOLO elements αποστέλλονται στο back-end, όπου πραγματοποιείται η κλήση προς το Gemini.

ΠΧ-5 (Προβολή αποτελεσμάτων): Μετά την επιστροφή του audit, τα issues που διαθέτουν έγκυρο anchor εμφανίζονται ως AR boxes επάνω στην οθόνη. Όσα issues δεν μπορούν να συνδεθούν με συγκεκριμένο στοιχείο παραμένουν διαθέσιμα στο UX Report panel και στην πλήρη αναφορά.

ΠΧ-6 (Άνοιγμα προηγούμενου audit): Από τη διαδρομή /audits, ο χρήστης επιλέγει μία παλαιότερη εγγραφή και μεταφέρεται στην αναλυτική αναφορά της. Τα αποθηκευμένα αποτελέσματα ανακτώνται από τη MySQL χωρίς να πραγματοποιείται νέα κλήση προς το Gemini.

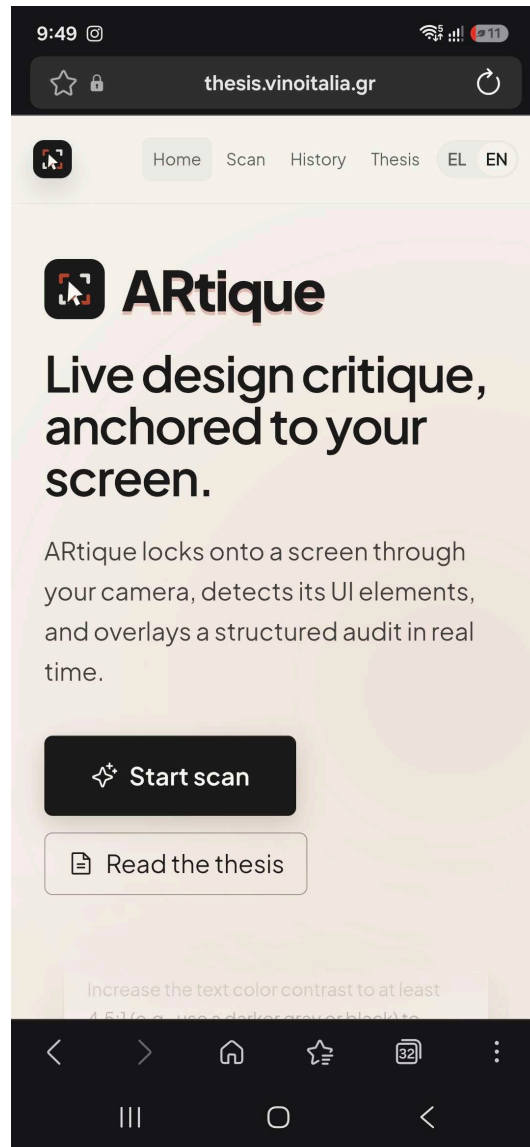
4.3.5.1 Αρχική Οθόνη

Η αρχική οθόνη έχει απλό ρόλο: παρουσιάζει την εφαρμογή και οδηγεί τον χρήστη στη σάρωση. Δεν περιλαμβάνει μεγάλο αριθμό επιλογών, επειδή ο βασικός στόχος είναι να ξεκινήσει γρήγορα η διαδικασία. Η ύπαρξη ξεκάθαρα κουμπιού για την έναρξη βοηθά ώστε ο χρήστης να καταλάβει ότι η κύρια λειτουργία της εφαρμογής είναι η αξιολόγηση μιας ορατής διεπαφής μέσω κάμερας. Στο Σχήμα 4.17 παρουσιάζεται αρχική σελίδα του ARtique σε επιτραπέζια προβολή. Στο Σχήμα 4.18 παρουσιάζεται αρχική σελίδα του ARtique σε πραγματική κινητή συσκευή. Στο Σχήμα 4.19 παρουσιάζεται επιλογή τρόπου σάρωσης σε επιτραπέζια προβολή. Στο Σχήμα 4.20 παρουσιάζεται επιλογή τρόπου σάρωσης σε πραγματική κινητή συσκευή. Στο Σχήμα 4.21 παρουσιάζεται επιλογή δοκιμαστικών ή πραγματικών σχολίων AI σε επιτραπέζια προβολή. Στο Σχήμα 4.22 παρουσιάζεται επιλογή λειτουργίας σχολίων σε πραγματική κινητή συσκευή. Στο Σχήμα 4.23 παρουσιάζεται πύλη εισαγωγής κωδικού για την πραγματική ανάλυση, χωρίς ανοικτό πληκτρολόγιο.



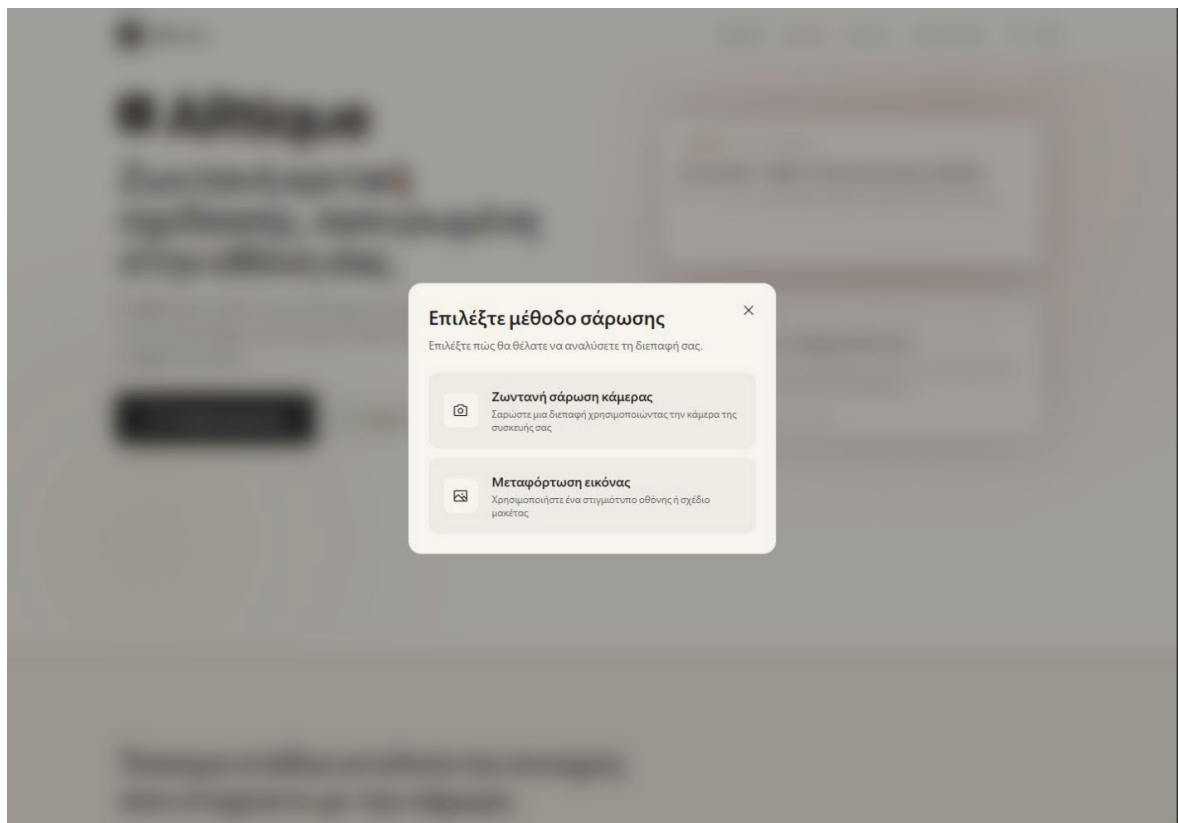
Σχήμα 4.17: Αρχική σελίδα του ARTique σε επιτραπέζια προβολή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARTique.



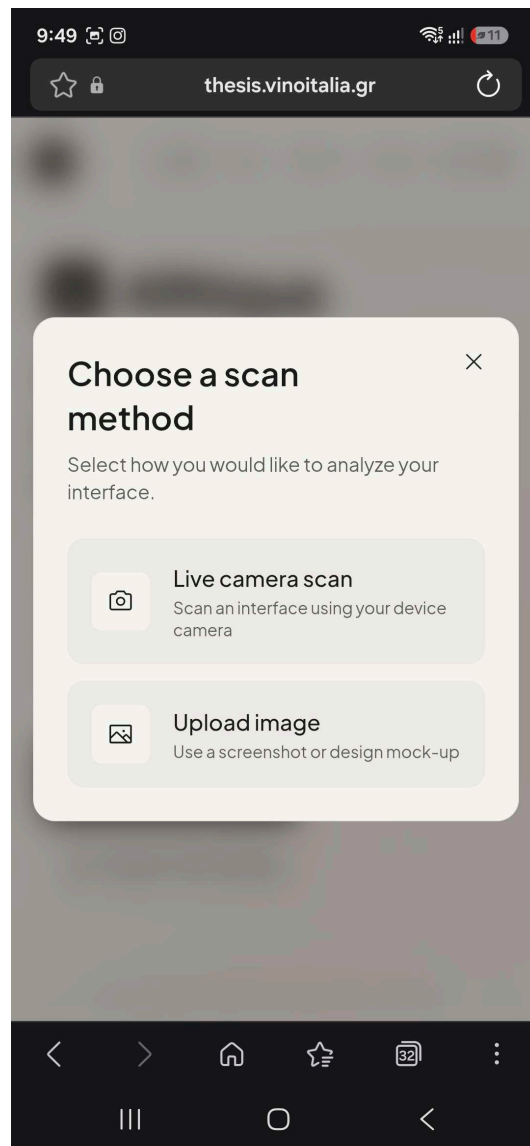
Σχήμα 4.18: Αρχική σελίδα του ARtique σε πραγματική κινητή συσκευή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



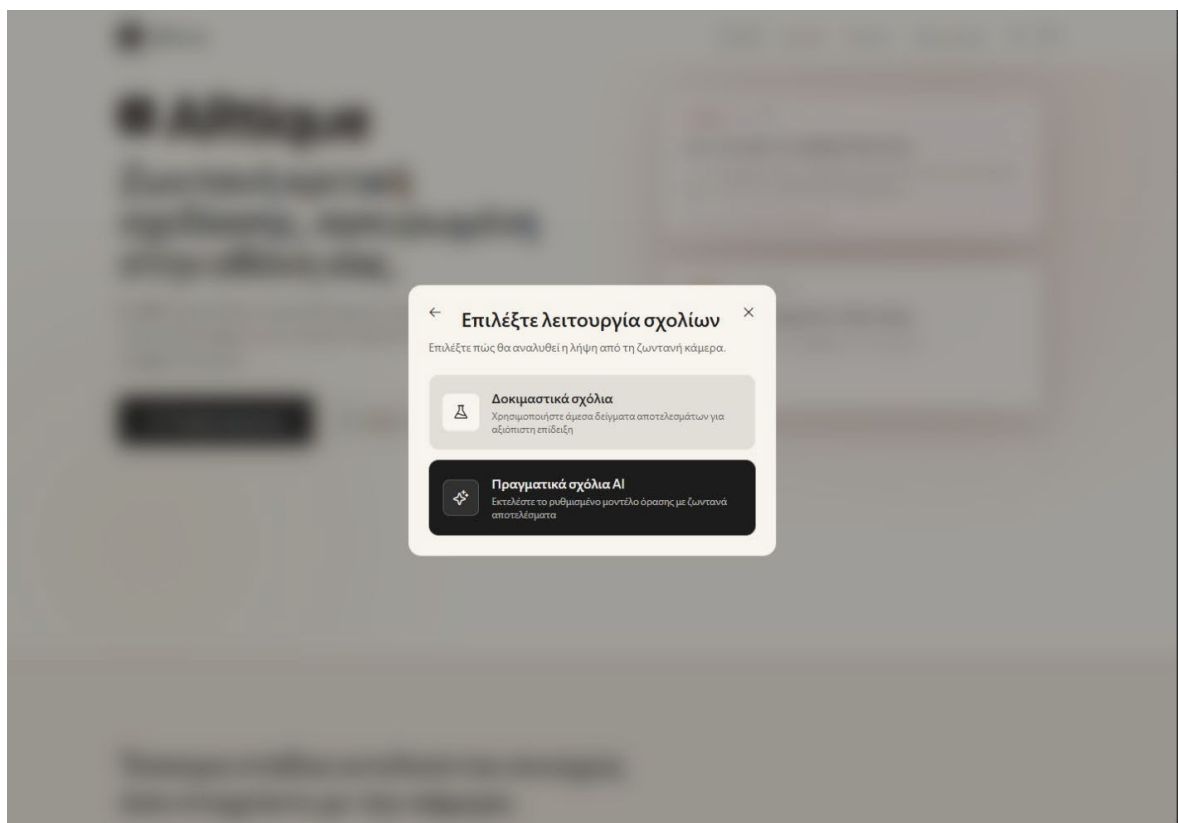
Σχήμα 4.19: Επιλογή τρόπου σάρωσης σε επιτραπέζια προβολή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



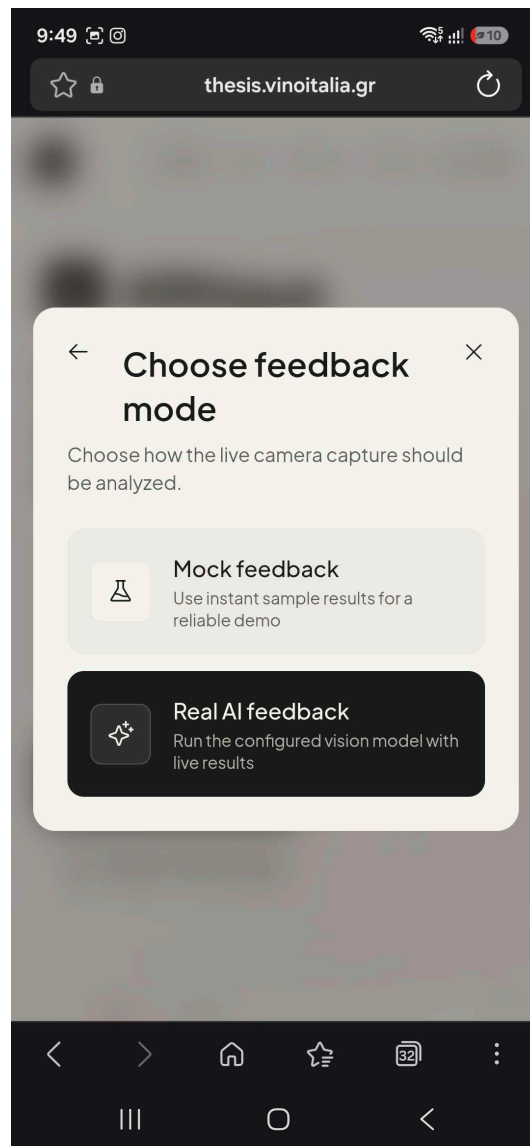
Σχήμα 4.20: Επιλογή τρόπου σάρωσης σε πραγματική κινητή συσκευή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



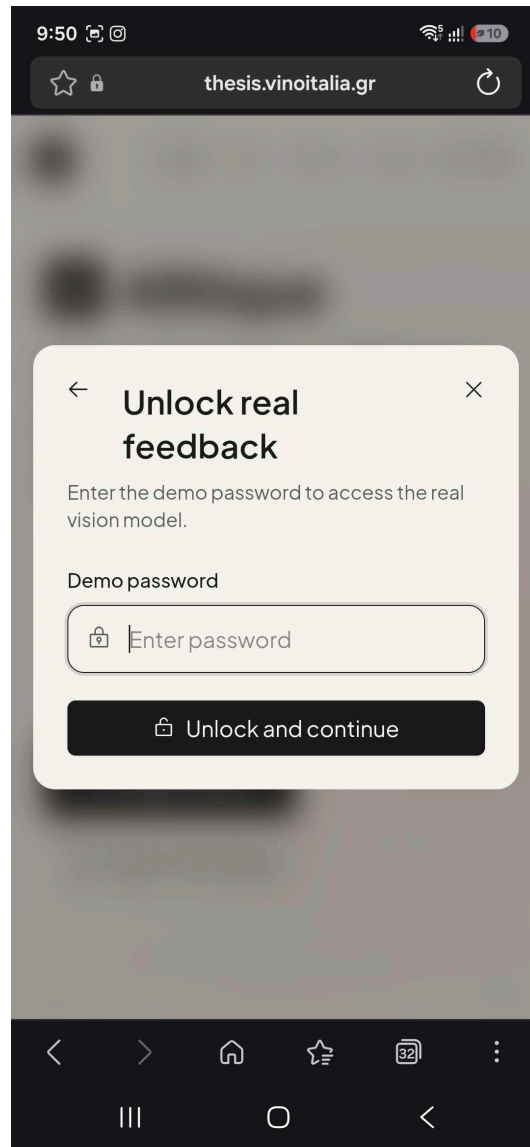
Σχήμα 4.21: Επιλογή δοκιμαστικών ή πραγματικών σχολίων AI σε επιτραπέζια προβολή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



Σχήμα 4.22: Επιλογή λειτουργίας σχολίων σε πραγματική κινητή συσκευή.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



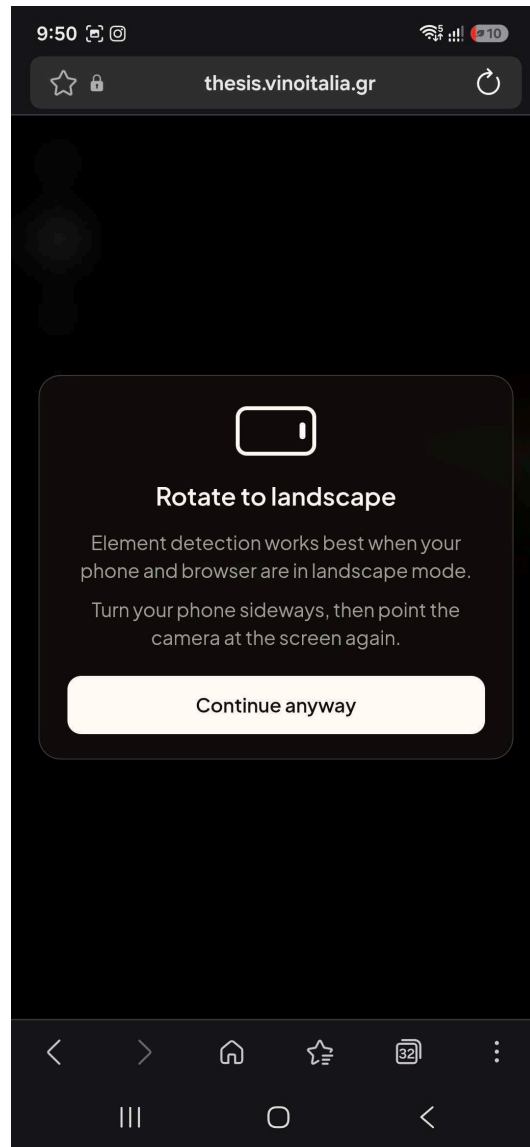
Σχήμα 4.23: Πύλη εισαγωγής κωδικού για την πραγματική ανάλυση, χωρίς ανοικτό πληκτρολόγιο.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARTique.

4.3.5.2 Οθόνη Σάρωσης

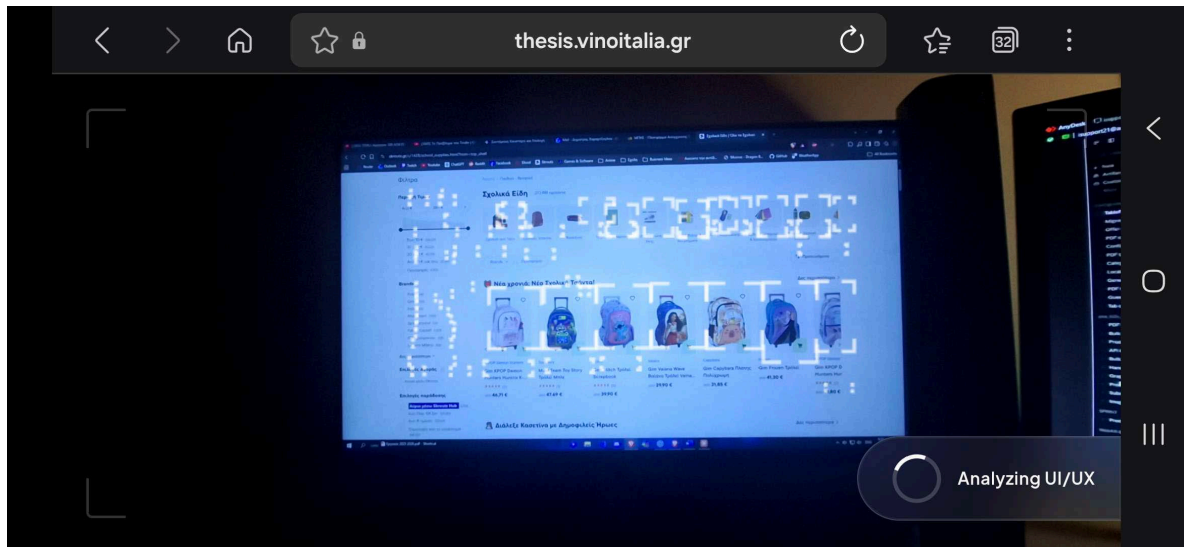
Η οθόνη σάρωσης είναι το πιο σημαντικό σημείο αλληλεπίδρασης. Εμφανίζει το live video, το περίγραμμα της ανιχνευμένης οθόνης, τις ενδείξεις των στοιχείων που έχουν βρεθεί και το κουμπί λήψης. Στην ίδια οθόνη εμφανίζονται επίσης μηνύματα κατάστασης, όπως αν η εφαρμογή εντοπίζει ακόμα στοιχεία ή αν είναι έτοιμη για σάρωση. Η πληροφορία αυτή δίνεται ως βοήθεια και όχι ως υποχρεωτικός περιορισμός.

Στην οθόνη scan τα βασικά controls παραμένουν στις άκρες του camera view, ώστε το screen quad και τα detected elements να παραμένουν ορατά. Σε landscape διάταξη το κουμπί capture μεταφέρεται στη δεξιά πλευρά. Η εφαρμογή εμφανίζει επίσης οδηγία περιστροφής, επειδή ο mobile browser μπορεί να αγνοήσει το αίτημα κλειδώματος προσανατολισμού και να δώσει camera frame με διαφορετική αναλογία. Στο Σχήμα 4.24 παρουσιάζεται οδηγία αλλαγής σε landscape προσανατολισμό πριν από τη ζωντανή σάρωση. Στο Σχήμα 4.25 παρουσιάζεται κατάσταση ανάλυσης UI/UX ενώ το live camera view παραμένει ενεργό.



Σχήμα 4.24: Οδηγία αλλαγής σε landscape προσανατολισμό πριν από τη ζωντανή σάρωση.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



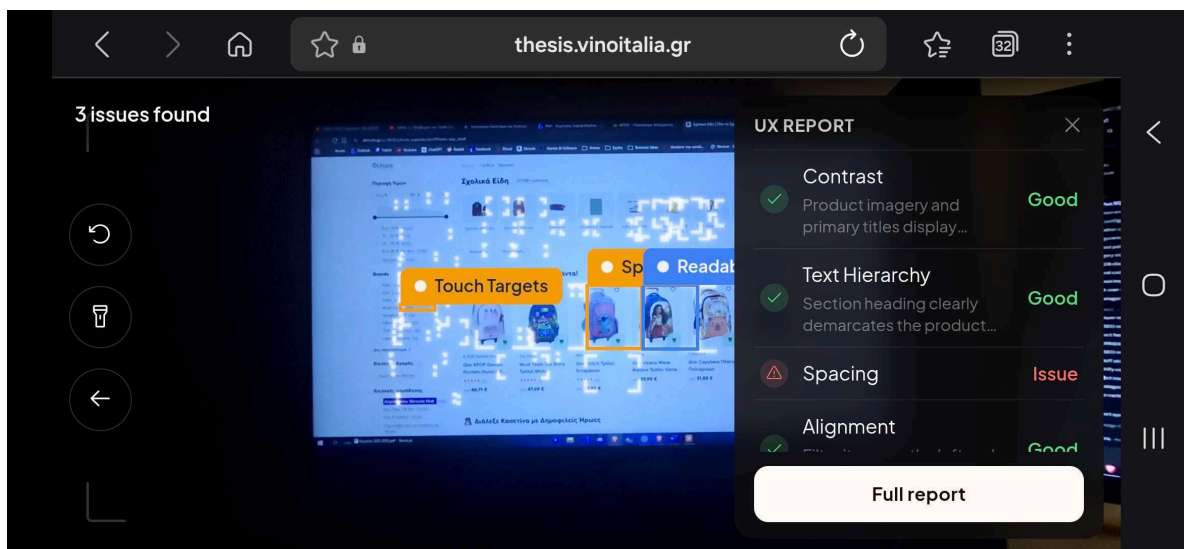
Σχήμα 4.25: Κατάσταση ανάλυσης UI/UX ενώ το live camera view παραμένει ενεργό.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.

4.3.5.3 Οθόνη Αποτελεσμάτων AR

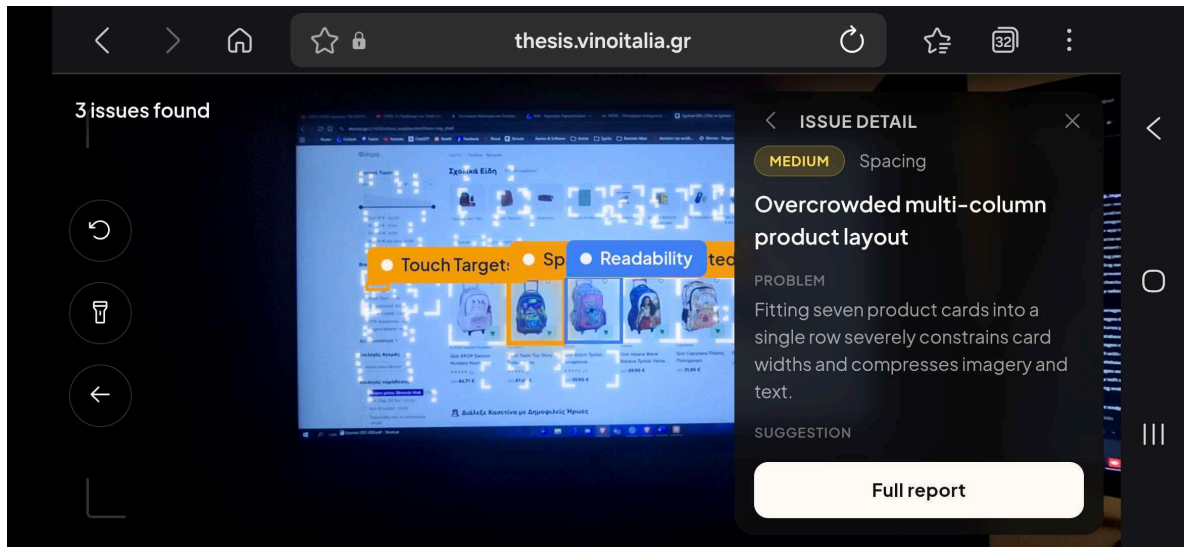
Το ar-marker-layer προβάλλει τις τέσσερις γωνίες κάθε anchored bounding box πάνω στο σταθεροποιημένο screen quad. Έτσι το πλαίσιο ακολουθεί την προοπτική της οθόνης και δεν σχεδιάζεται απλώς σε συντεταγμένες του viewport. Με την επιλογή ενός issue, το αντίστοιχο πλαίσιο και η ετικέτα του περνούν σε ενεργή κατάσταση.

Το UX Report panel εμφανίζει τις έντεκα κατηγορίες του checklist ως Good ή Issue. Οι συναρτήσεις normalizeChecklist στο back-end και deriveChecklist στο front-end μετατρέπουν σε Issue κάθε κατηγορία που περιέχει πρόβλημα, ώστε το panel να συμφωνεί με τα AR boxes. Στο Σχήμα 4.26 παρουσιάζεται προβολή του UX Report με τις καταστάσεις Good και Issue. Στο Σχήμα 4.27 παρουσιάζεται λεπτομέρεια επιλεγμένου issue με κατηγορία, περιγραφή και πρόταση βελτίωσης.



Σχήμα 4.26: Προβολή του UX Report με τις καταστάσεις Good και Issue.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.



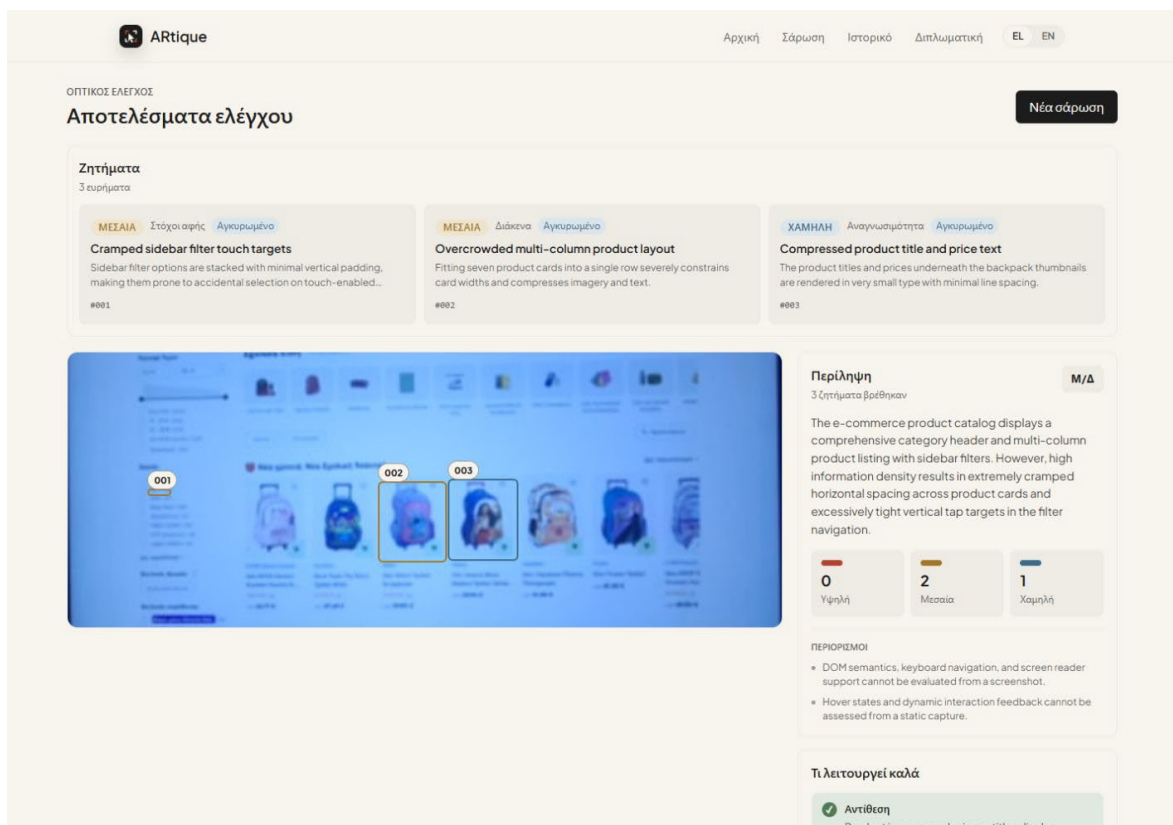
Σχήμα 4.27: Λεπτομέρεια επιλεγμένου issue με κατηγορία, περιγραφή και πρόταση βελτίωσης.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARtique.

4.3.5.4 Αναλυτική Αναφορά και Ιστορικό Audits

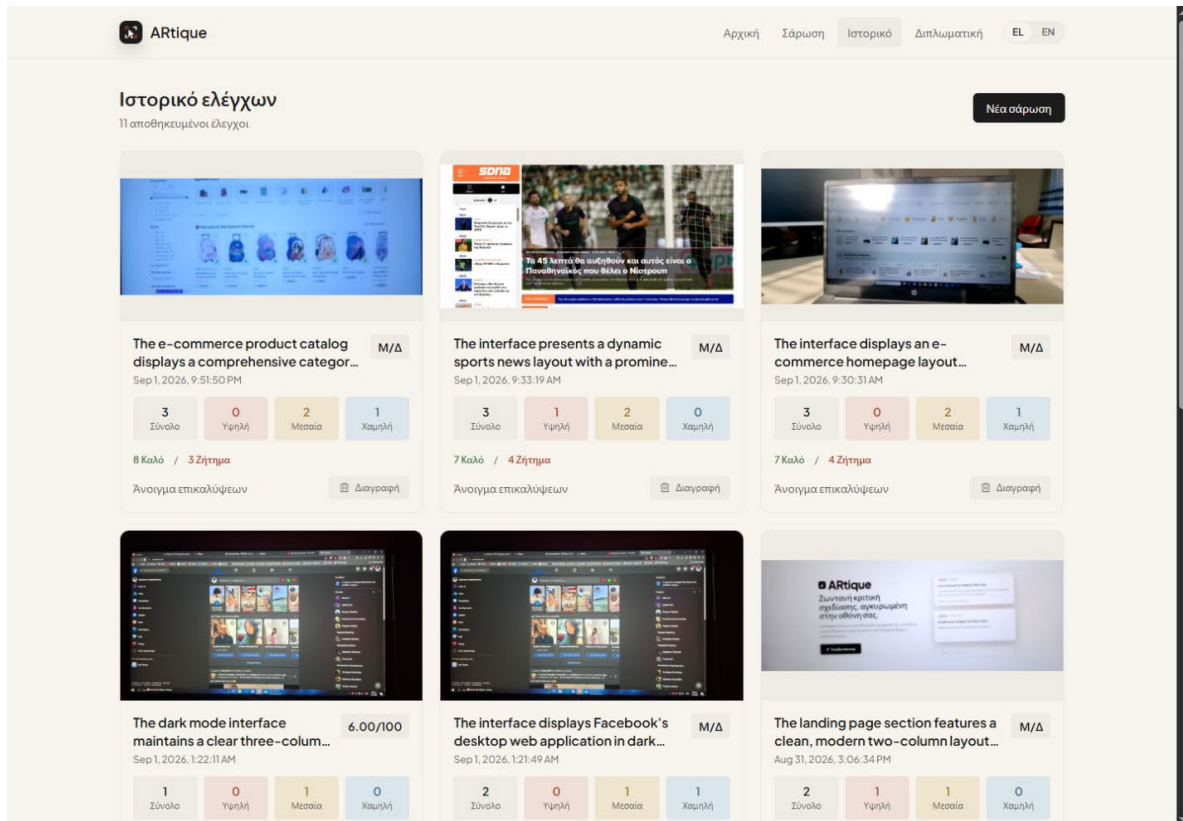
Από τη διαδρομή /audit/:id ανοίγει ένα συγκεκριμένο audit, το οποίο αναζητείται με το public id του. Στην οθόνη εμφανίζονται η σύνοψη και τα issues που έχουν αποθηκευτεί. Κάθε issue συνοδεύεται από την κατηγορία, τη σοβαρότητα, το confidence, την περιγραφή και την προτεινόμενη βελτίωση. Όσα issues έχουν χαρακτηριστεί ως report-only παραμένουν στην αναφορά, αλλά δεν προβάλλονται ως πλαίσια στην εικόνα AR.

Το ιστορικό των αναλύσεων βρίσκεται στη διαδρομή /audits. Η σελίδα ζητά από το API τις εγγραφές της MySQL και, όταν επιλεγεί μία από αυτές, ανοίγει το αντίστοιχο audit. Δεν πραγματοποιείται νέα κλήση προς το Gemini, επειδή χρησιμοποιούνται τα ήδη αποθηκευμένα αποτελέσματα και issues. Η εικόνα της αρχικής λήψης δεν διατηρείται στη βάση, επομένως επαναφορτώνεται μόνο η αναφορά. Στο Σχήμα 4.28 παρουσιάζεται πλήρης αναφορά πραγματικού audit με αγκυρωμένα ευρήματα και σύνοψη. Στο Σχήμα 4.29 παρουσιάζεται φορτωμένο ιστορικό αποθηκευμένων audits με προεπισκοπήσεις και κατανομή ζητημάτων.



Σχήμα 4.28: Πλήρης αναφορά πραγματικού audit με αγκυρωμένα ευρήματα και σύνοψη.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARTique.



Σχήμα 4.29: Φορτωμένο ιστορικό αποθηκευμένων audits με προεπισκοπήσεις και κατανομή ζητημάτων.

Πηγή: Προσωπική λήψη από την υλοποίηση του ARTique.

4.4 Έλεγχοι συνέπειας και περιορισμοί υλοποίησης

Η συνέπεια της υλοποίησης βασίζεται στη διατήρηση κοινής χωρικής αναφοράς από το screen quad έως τα AR overlays. Το ίδιο πλαίσιο χρησιμοποιείται για το crop που λαμβάνει το YOLO, για τα tracks των στοιχείων και για την προβολή των anchors. Στο back-end, τα uiElementsJson ελέγχονται πριν από τη σύνθεση της αναφοράς, ώστε η σύνδεση ενός issue με elementId να επιβεβαιώνεται ή να διορθώνεται με overlap και γειτονικότητα όπου αυτό είναι τεκμηριωμένο.

Οι έλεγχοι αυτοί δεν μετατρέπουν το σύστημα σε πλήρη μηχανισμό αξιολόγησης διεπαφών. Η σταθερότητα των tracks επηρεάζεται από την κάμερα, τη γωνία λήψης, τον φωτισμό, το dataset και τις δυνατότητες της συσκευής. Παράλληλα, η οπτική ανάλυση δεν τεκμηριώνει ιδιότητες του DOM, ARIA, πλοήγηση με πληκτρολόγιο ή συμπεριφορά screen reader. Οι περιορισμοί αυτοί ορίζουν με σαφήνεια το πεδίο στο οποίο μπορούν να ερμηνευθούν τα αποτελέσματα της παρούσας υλοποίησης.

4.5 Επίλογος

Στην πράξη, η εφαρμογή εκτελεί δύο διαφορετικές διαδικασίες. Όσο η κάμερα παραμένει ανοιχτή, το Angular εντοπίζει την οθόνη, σταθεροποιεί το screen quad, εκτελεί το YOLO και ενημερώνει τα overlays. Όταν ο χρήστης πατήσει capture, αποστέλλονται στο back-end ένα στιγμιότυπο της οθόνης και τα YOLO elements που ήταν διαθέσιμα εκείνη τη στιγμή. Το back-end ζητά την αξιολόγηση από το Gemini, ελέγχει την απάντηση και αποθηκεύει το audit στη MySQL.

Οι ρόλοι των επιμέρους μηχανισμών παραμένουν διακριτοί. Το screen detection εντοπίζει την οθόνη και το YOLO βρίσκει τα ορατά στοιχεία της διεπαφής μαζί με τις θέσεις τους. Κανένα από τα δύο δεν αποφασίζει αν υπάρχει σχεδιαστικό πρόβλημα. Αυτή η αξιολόγηση γίνεται από το Gemini, το οποίο επιστρέφει τα issues σε συγκεκριμένη μορφή JSON και τα συνδέει, όπου είναι δυνατό, με τα διαθέσιμα anchors. Το αποτέλεσμα αφορά όσα μπορούν να εκτιμηθούν από την εικόνα και λειτουργεί ως υποστήριξη του ανθρώπινου ελέγχου, όχι ως πλήρης αξιολόγηση προσβασιμότητας.

Κεφάλαιο 5: Συμπεράσματα και Μελλοντική Εργασία

5.1 Εισαγωγή

Η ενότητα εξετάζει την υλοποίηση του ARtique ως ερευνητικού πρωτοτύπου. Δεν εκδίδει συνολική κρίση για μια ιστοσελίδα. Παράγει έναν αρχικό οπτικό έλεγχο για τα στοιχεία που βρίσκονται μέσα στο κάδρο.

Η διαδικασία αρχίζει με το ζωντανό feed της κάμερας. Στο Angular, η ανίχνευση της οθόνης γίνεται με OpenCV.js. Ακολουθεί η σταθεροποίηση του screen quad. Το YOLO περιορίζεται στο crop της διεπαφής. Έξω από αυτό το crop, το καρέ δεν αναλύεται.

Στο συγκεκριμένο πρωτότυπο, το YOLO και το Gemini μπαίνουν σε διαφορετικά σημεία της διαδικασίας. Από το YOLO προκύπτουν τα bounding boxes και τα tracks των στοιχείων που φαίνονται στην εικόνα. Την οπτική αξιολόγηση UI/UX τη διατυπώνει το Gemini. Το bounding box εντοπίζει τη θέση ενός στοιχείου· δεν δείχνει αν υπάρχει σχεδιαστικό πρόβλημα. Μια παρατήρηση από το Gemini μπορεί να υπάρχει στην αναφορά. Μια παρατήρηση χρειάζεται αντιστοίχιση με συγκεκριμένο στοιχείο της διεπαφής, ώστε να εμφανιστεί ως AR overlay.

Κατά το capture, μαζί με την εικόνα στέλνονται τα uiElementsJson που είναι διαθέσιμα εκείνη τη στιγμή. Πριν φτάσουν στο anchor-first prompt, τα δεδομένα ελέγχονται στο back-end. Όταν το back-end λαμβάνει issues, ελέγχει την αντιστοίχισή τους με τα διαθέσιμα στοιχεία. Η αντιστοίχιση διορθώνεται, όπου χρειάζεται, με βάση την επικάλυψη και τη γειτνίαση. Η αναφορά μπορεί να έχει report-only παρατηρήσεις. Χωρίς αξιόπιστο anchor, δεν προβάλλονται ως AR marker.

Οι περιορισμοί του πρωτοτύπου καθορίζουν και το εύρος όσων μπορούν να συναχθούν από τα αποτελέσματα. Η εφαρμογή αναλύει ό,τι εμφανίζεται στην εικόνα της κάμερας. DOM, ARIA attributes, πλοήγηση με πληκτρολόγιο και συμπεριφορά screen reader μένουν εκτός αυτού του ελέγχου. Το live YOLO επηρεάζεται από το dataset, τον φωτισμό, τη γωνία λήψης και την απόδοση της συσκευής. Η σταθεροποίηση συγκρατεί τις μετακινήσεις των overlays, αλλά δεν αρκεί για αξιολόγηση σε πραγματικές συνθήκες χρήσης.

Η MySQL επιτρέπει να ανακτηθεί ένα προηγούμενο audit χωρίς νέα κλήση στο μοντέλο. Αποθηκεύονται τα μεταδεδομένα του audit, τα issues και το raw AI JSON. Στην παρούσα υλοποίηση, τα buffers των εικόνων δεν κρατούνται μόνιμα. Έτσι, δεν διατηρείται μόνιμα το οπτικό υλικό. Τα μελλοντικά test cases και τα τεκμήριά τους χρειάζονται χωριστή οργάνωση.

Η αξιολόγηση χρειάζεται να διακρίνει την ποιότητα της UI/UX κριτικής από την ποιότητα της χωρικής αγκύρωσης. Ένα issue μπορεί να είναι χρήσιμο στο UI/UX review και παρ' όλα αυτά να μην διαθέτει anchor αρκετά σταθερό για προβολή στο live screen quad. Το ότι το YOLO εντοπίζει σωστά ένα στοιχείο δεν αρκεί για να θεωρηθεί προβληματική η διεπαφή. Σε αυτή τη φάση, η σύγκριση με

ανθρώπινο review και η καταγραφή feedback μπορούν να χρησιμοποιηθούν για ακρίβεια, χρησιμότητα και συμφωνία ως προς τη σοβαρότητα. Δεν πρόκειται για αποτελέσματα που έχουν ήδη μετρηθεί.

Στις επόμενες ενότητες καταγράφονται τα συμπεράσματα της υλοποίησης και όσα μένουν για τη συνέχεια της εργασίας. Η συζήτηση στηρίζεται σε όσα υλοποιήθηκαν και μπορούν να τεκμηριωθούν από τα διαθέσιμα στοιχεία. Το πρωτότυπο δεν παρουσιάζεται ως μηχανισμός πιστοποίησης προσβασιμότητας ούτε ως υποκατάστατο του ανθρώπινου UI/UX review.

5.2 Πλαίσιο ερμηνείας των αποτελεσμάτων

Στο audit διατηρούνται χωριστά η κριτική της εικόνας και η θέση του στοιχείου. Το Gemini περιγράφει το issue και το κατατάσσει σε κατηγορία. Στην ίδια απάντηση περιλαμβάνονται η σοβαρότητα και η προτεινόμενη βελτίωση. Αν υπάρχει διαθέσιμο track, το back-end ελέγχει αν μπορεί να χρησιμοποιηθεί για την προβολή. Το κείμενο του issue και το anchor συνδέονται, χωρίς να είναι το ίδιο δεδομένο.

Στο UX Report τα ευρήματα μπαίνουν στις έντεκα σταθερές κατηγορίες: contrast, hierarchy, spacing, alignment, touch-target, cta, readability, clutter, navigation, consistency και accessibility. Η ένδειξη Issue μπαίνει σε κατηγορία όταν υπάρχει σχετικό εύρημα. Με αυτόν τον τρόπο, η σύνοψη περιλαμβάνει τα ίδια issues με την αναφορά. Η ένδειξη βοηθά στην πλοήγηση μέσα στο αποτέλεσμα. Δεν δίνει βαθμό ποιότητας για την ιστοσελίδα.

Το overallScore μένει null στην τρέχουσα υλοποίηση. Μέθοδος υπολογισμού για το πεδίο δεν έχει οριστεί ακόμη και δεν υπάρχουν δεδομένα αξιολόγησης για να στηριχθεί. Το overallScore μένει λοιπόν έξω από την αριθμητική εκτίμηση της ποιότητας μιας ιστοσελίδας. Λίγες οπτικές παρατηρήσεις δεν επαρκούν για μια σταθερή γενική τιμή που να έχει νόημα σε διαφορετικές διεπαφές.

Όταν ένα issue περιλαμβάνει elementId, το bounding box λαμβάνεται από το αντίστοιχο στοιχείο. Η θέση δεν επανυπολογίζεται από το γλωσσικό μοντέλο. Αν το back-end δεν επιβεβαιώσει τη σύνδεση, αναζητά overlap ή κοντινό στοιχείο και τη διορθώνει. Όταν δεν βρίσκεται λογικό anchor, το issue μένει μόνο στο report. Αποφεύγεται έτσι ένα AR πλαίσιο σε άσχετη θέση. Μια τέτοια προβολή θα μπορούσε να παραπλανήσει περισσότερο από μια γραπτή παρατήρηση.

Η κίνηση της εικόνας επηρεάζει και τη σταθερότητα της προβολής. Το screen quad λειτουργεί ως περίγραμμα της οθόνης. Με βάση αυτό ορίζεται το crop του YOLO και η θέση των AR overlays. Το deadband απορροφά μικρές αλλαγές στη θέση. Μια απότομη μετακίνηση εφαρμόζεται αφού επιβεβαιωθεί. Όταν η κάμερα παραμένει ακίνητη, η θέση παγώνει. Οι ρυθμίσεις αυτές αφορούν τη σταθερότητα της προβολής, όχι την ακρίβεια του detector. Τα boxes κρατούν τη θέση τους όταν η εικόνα έχει μικρές διακυμάνσεις.

Η χρήση του ARtique παραμένει υπό την κρίση του χρήστη. Τα anchors χρησιμοποιούνται για να βρεθεί το ορατό στοιχείο στο οποίο αναφέρεται η πρόταση. Στο report εξηγείται η παρατήρηση που καταγράφηκε. Ο σχεδιαστής ή ο αξιολογητής αποφασίζει πόσο σημαντικό είναι το issue και κατά πόσο ταιριάζει η πρόταση στο προϊόν. Σε αυτόν ανήκει και η απόφαση για τυχόν έλεγχο προσβασιμότητας πέρα από όσα φαίνονται στην εικόνα.

5.3 Συμπεράσματα

Στην εργασία αυτή υλοποιήθηκε το ARtique, μια web εφαρμογή που εξετάζει μια ορατή διεπαφή χρήστη μέσα από την κάμερα της συσκευής. Η διαδικασία δεν είναι απλή αποστολή μιας εικόνας σε AI μοντέλο. Το σύστημα ξεκινά από το camera feed, εντοπίζει την οθόνη, δημιουργεί audit και εμφανίζει τις παρατηρήσεις πάνω στη διεπαφή.

Η αναφορά χρειάζεται να δείχνει σε ποιο σημείο της οθόνης αναφέρεται. Διαφορετικά, σε μια παρατήρηση για αντίθεση ή ευθυγράμμιση, ο χρήστης θα έπρεπε να εντοπίσει μόνος του το σχετικό στοιχείο. Γι' αυτό στο ARtique τα προβλήματα δεν εμφανίζονται μόνο ως λίστα, αλλά προσπαθούν να συνδεθούν με συγκεκριμένες περιοχές της οθόνης. Αυτή η σύνδεση δίνει στην εφαρμογή τον Web AR χαρακτήρα της.

Κάθε τεχνολογία καλύπτει διαφορετικό μέρος της ροής. Το screen detection εντοπίζει την οθόνη μέσα στο live video και επιστρέφει το screen quad. Το YOLO εντοπίζει ορατά στοιχεία της διεπαφής και επιστρέφει bounding boxes. Το Gemini κάνει την οπτική αξιολόγηση. Τα bounding boxes του YOLO χρησιμοποιούνται ώστε οι παρατηρήσεις να συνδεθούν με σημεία που φαίνονται στην οθόνη.

Η εικόνα και τα στοιχεία που εντοπίζει το YOLO στέλνονται στο back-end ως multipart/form-data. Πριν γίνει η κλήση προς το μοντέλο, η εικόνα συμπιέζεται και προετοιμάζεται για αποστολή. Με αυτόν τον τρόπο μειώνεται ο όγκος του request.

Το audit δεν αποθηκεύεται αμέσως μόλις επιστρέψει από το μοντέλο. Ελέγχονται πρώτα οι κατηγορίες, το checklist και τα anchors. Αν ένα issue δεν μπορεί να συνδεθεί με σιγουριά με κάποιο ορατό στοιχείο, κρατιέται στην αναφορά, αλλά δεν εμφανίζεται ως AR marker. Διαφορετικά, ο χρήστης θα μπορούσε να συνδέσει την παρατήρηση με άσχετο στοιχείο της διεπαφής.

Τα audits αποθηκεύονται στη MySQL και μπορούν να ανακτηθούν από το ιστορικό, χωρίς νέα κλήση προς το Gemini. Το back-end διαθέτει επίσης endpoint για feedback. Η αντίστοιχη λειτουργία στο front-end δεν έχει ολοκληρωθεί. Σε επόμενο βήμα, μπορεί να χρησιμοποιηθεί για την καταγραφή της γνώμης του χρήστη σχετικά με τη χρησιμότητα, την ακρίβεια και τη σοβαρότητα κάθε issue.

Ένας βασικός περιορισμός είναι ότι το ARtique βλέπει την ιστοσελίδα ως εικόνα. Δεν έχει πρόσβαση στο DOM, στα ARIA attributes, στην πλοήγηση με πληκτρολόγιο ή στη συμπεριφορά screen readers. Επομένως, δεν μπορεί να παρουσιαστεί ως πλήρης έλεγχος προσβασιμότητας ή ως WCAG πιστοποίηση. Μπορεί να εντοπίσει μόνο προβλήματα που φαίνονται οπτικά, όπως αντίθεση, αναγνωσιμότητα, spacing, alignment, touch targets και σαφήνεια βασικών ενεργειών. [2], [4]

Υπάρχουν επίσης περιορισμοί στο live YOLO pipeline. Η ποιότητα των bounding boxes εξαρτάται από το dataset και από τις συνθήκες λήψης. Ο φωτισμός, η γωνία της κάμερας, οι αντανακλάσεις και η απόδοση της συσκευής επηρεάζουν το αποτέλεσμα. Οι μηχανισμοί σταθεροποίησης, όπως το deadband και το πάγωμα του screen quad όταν η κάμερα μένει ακίνητη, βελτιώνουν την εικόνα των overlays. Δεν αρκούν όμως από μόνοι τους για να καλύψουν την ανάγκη για καλύτερο μοντέλο και περισσότερες δοκιμές σε πραγματικές συσκευές.

Το ARtique μπορεί να δώσει μια αρχική οπτική ένδειξη για θέματα UI/UX. Δεν εκδίδει απόφαση για την ποιότητα μιας ιστοσελίδας και δεν υποκαθιστά τον έλεγχο από ειδικό. Στην τρέχουσα μορφή του, μπορεί να λειτουργήσει ως πρώτο φίλτρο: ο προγραμματιστής ή ο σχεδιαστής βλέπει τα πιθανά ζητήματα και εξετάζει στη συνέχεια όσα θεωρεί σχετικά.

5.4 Μελλοντική εργασία και βελτιστοποίηση

Το σημείο που χρειάζεται περισσότερη διερεύνηση στο ARtique είναι το μοντέλο YOLO. Η αρχική εκπαίδευση έγινε με περιορισμένο σύνολο δεδομένων, επομένως η συμπεριφορά του μοντέλου δεν είναι ίδια σε όλες τις συνθήκες λήψης. Ο φωτισμός, η γωνία της κάμερας, οι αντανakλάσεις και η ποιότητα της συσκευής μπορούν να επηρεάσουν τον εντοπισμό των στοιχείων της διεπαφής.

Η βελτίωση του dataset είναι το επόμενο βήμα. Χρειάζεται υλικό από πραγματικές λήψεις κινητών σε διαφορετικές οθόνες και περιβάλλοντα. Το σύνολο εκπαίδευσης πρέπει να παραμείνει διακριτό από το σύνολο ελέγχου. Χρειάζεται επίσης καλύτερη κατανομή των UI elements ανά κατηγορία. Υπάρχει ήδη camera-domain augmentation. Απομένει να εκπαιδευτεί ξανά το μοντέλο με τα εμπλουτισμένα δεδομένα και να ελεγχθεί σε ανεξάρτητες εικόνες. Έπειτα θα μπορεί να ελεγχθεί αν τα bounding boxes πέφτουν στα στοιχεία που προορίζεται να βρίσκει το μοντέλο.

Η απόδοση στα κινητά αποτελεί επίσης θέμα για μελλοντική εργασία. Όπου το υποστηρίζει ο browser, η προεπεξεργασία και το inference μεταφέρονται σε Web Worker με OffscreenCanvas. Παρ' όλα αυτά, η συμπεριφορά της εφαρμογής χρειάζεται έλεγχο σε πραγματικές συσκευές. Η τρέχουσα ταχύτητα του live YOLO δεν καλύπτει όλες τις περιπτώσεις χρήσης. Η καθυστέρηση μπορεί να επηρεάσει την AR εμπειρία, ακόμη κι όταν το screen quad και τα overlays μένουν σταθερά.

Χρειάζεται ακόμη σύγκριση ανάμεσα στις εκδόσεις του μοντέλου με είσοδο 640 και 1024 pixels. Η μεγαλύτερη ανάλυση μπορεί να βελτιώνει την αναγνώριση μικρών στοιχείων, αλλά απαιτεί περισσότερους πόρους. Η απόφαση χρειάζεται να συνεκτιμήσει την ποιότητα των detections και τον ρυθμό ανανέωσης. Αυτό χρειάζεται να δοκιμαστεί σε κινητές συσκευές, ώστε να προκύψει ο κατάλληλος συνδυασμός ακρίβειας και απόδοσης για την εφαρμογή.

Η τελική μεθοδολογία αξιολόγησης του ARtique δεν έχει ακόμη οριστεί. Πριν ξεκινήσει η αξιολόγηση, πρέπει να οριστεί αν θα γίνουν δοκιμές σε πραγματικές ιστοσελίδες, σε ελεγχόμενα παραδείγματα ή και στα δύο. Για κάθε audit μπορεί να γίνεται αξιολόγηση από άτομο με γνώση UI/UX. Η αξιολόγηση μπορεί να ελέγχει αν το issue είναι σωστό και αν η σοβαρότητά του είναι εύλογη. Χρειάζεται επίσης να καταγράφει αν η οπτική επισήμανση εμφανίζεται στη σωστή περιοχή της οθόνης.

Υπάρχει ήδη στο back-end endpoint feedback, με δυνατότητα αποθήκευσης αξιολογήσεων για χρησιμότητα, ακρίβεια και συμφωνία ως προς τη σοβαρότητα κάθε issue. Το Angular δεν έχει ακόμη αντίστοιχο χειρισμό στη διεπαφή. Με την προσθήκη του, μπορούν να συγκεντρωθούν αξιολογήσεις που θα χρησιμεύσουν στην επόμενη φάση ελέγχου του συστήματος.

Το πεδίο overallScore παραμένει null. Το πεδίο υπάρχει στο schema, χωρίς να χρησιμοποιείται ως πραγματική βαθμολογία. Μια μελλοντική βαθμολογία θα πρέπει να στηρίζεται σε αποτελέσματα αξιολόγησης και όχι μόνο σε έναν αριθμό του μοντέλου. Η ακρίβεια των issues και η χρησιμότητα των προτάσεων είναι πιθανά κριτήρια. Μπορεί να ληφθεί υπόψη και η συμφωνία με ανθρώπινη αξιολόγηση.

Τέλος, ορισμένοι περιορισμοί δεν αντιμετωπίζονται με βελτίωση του YOLO ή με περισσότερα δεδομένα κάμερας. Η ανάλυση του ARtique περιορίζεται σε όσα φαίνονται στην οθόνη. DOM, ARIA attributes και πλοήγηση με πληκτρολόγιο μένουν έξω από το πεδίο του. Σε μελλοντική έκδοση θα μπορούσε να προστεθεί ένας ξεχωριστός έλεγχος του κώδικα της ιστοσελίδας. Ο οπτικός έλεγχος και ο έλεγχος κώδικα θα πρέπει να παρουσιάζονται ως διαφορετικές λειτουργίες, ώστε να είναι σαφές από πού προέρχεται κάθε εύρημα.

Για την επόμενη αξιολόγηση χρειάζεται ένα επαναλήψιμο σύνολο δοκιμών. Τα παραδείγματα διεπαφών πρέπει να εξετάζονται υπό γνωστές συνθήκες λήψης. Για κάθε δοκιμή χρειάζεται καταγραφή της

συσκευής, του προσανατολισμού, της απόστασης και του φωτισμού. Η χρήση μόνο καθαρών screenshots δεν αρκεί για το συγκεκριμένο πρωτότυπο, επειδή το σύστημα λειτουργεί πάνω σε εικόνα από κάμερα και επηρεάζεται από προοπτική, glare, θόρυβο και blur. Τα camera-domain augmentations προετοιμάζουν το dataset. Για την αξιολόγηση χρειάζονται και mobile frames που δεν χρησιμοποιήθηκαν στην εκπαίδευση.

Η αξιολόγηση μπορεί να χωριστεί σε δύο επίπεδα. Στο πρώτο επίπεδο ελέγχεται αν ο detector βρίσκει στοιχεία κατάλληλα για anchors. Ελέγχεται επίσης αν οι projected περιοχές παραμένουν σταθερές όσο γίνεται η σάρωση. Στο δεύτερο εξετάζεται αν τα issues του audit έχουν νόημα για έναν ανθρώπινο αξιολογητή και αν οι προτάσεις είναι εφαρμόσιμες. Έτσι, η αξιολόγηση του YOLO δεν συγχέεται με την αξιολόγηση της κριτικής του Gemini.

Υπάρχει endpoint feedback, χωρίς πλήρη ενσωμάτωση της συλλογής αξιολογήσεων στη διεπαφή. Σε μελλοντική έκδοση, οι αξιολογητές θα μπορούν να δηλώνουν αν ένα issue είναι χρήσιμο και αν είναι ακριβές. Θα μπορούν επίσης να σχολιάζουν τη σοβαρότητά του. Η αποθήκευση αυτών των απαντήσεων μαζί με την αναφορά του audit θα δημιουργούσε ένα σαφές υλικό για σύγκριση με ανθρώπινο review και για τεκμηριωμένη συζήτηση των false positives ή των παραλείψεων.

Χρειάζεται επίσης έλεγχος της συμπεριφοράς σε διαφορετικές κινητές συσκευές. Η διαδρομή με Web Worker και OffscreenCanvas χρησιμοποιείται όπου υποστηρίζεται, όμως η πραγματική απόδοση εξαρτάται από τον browser και το διαθέσιμο hardware. Για τον λόγο αυτό, οι μετρήσεις δεν θα πρέπει να περιορίζονται στον ρυθμό inference. Κατά τη δοκιμή χρειάζεται να καταγράφεται πότε το screen quad παραμένει σταθερό. Χρειάζεται επίσης να σημειώνεται πότε δεν εμφανίζονται μη επιβεβαιωμένα tracks και πότε η καθυστέρηση δυσκολεύει την εμπειρία.

5.5 Επίλογος

Η υλοποίηση αποδεικνύει ότι η πλήρης ροή από τη ζωντανή λήψη έως την αποθήκευση και την επαναπροβολή ενός audit μπορεί να λειτουργήσει σε ενιαία web εφαρμογή. Στην υλοποίηση, η οπτική αξιολόγηση συνδέεται με anchors από την ανίχνευση στοιχείων. Οι παρατηρήσεις προβάλλονται πάνω στο σταθεροποιημένο screen quad.

Το αποτέλεσμα παραμένει ερευνητικό πρωτότυπο και όχι αυτόματος κριτής σχεδίασης ή εργαλείο πιστοποίησης προσβασιμότητας. Χρειάζεται βελτίωση του dataset του YOLO και έλεγχος της απόδοσης σε πραγματικές συσκευές.

Βιβλιογραφία

- [1] P. Duan, C.-Y. Cheng, G. Li, B. Hartmann, and Y. Li, “UICrit: Enhancing automated design evaluation with a UI critique dataset,” in Proc. 37th Annu. ACM Symp. User Interface Softw. Technol. (UIST ’24), 2024, doi: 10.1145/3654777.3676381.
- [2] World Wide Web Consortium, “Web Content Accessibility Guidelines (WCAG) 2.2,” W3C Recommendation, Dec. 2024. [Online]. Available: <https://www.w3.org/TR/WCAG22/>. [Accessed: Aug. 22, 2026].
- [3] P. Duan, C.-Y. Chen, B. Hartmann, and Y. Li, “Visual prompting with iterative refinement for design critique generation,” arXiv:2412.16829, 2024.
- [4] World Wide Web Consortium, “Accessible Rich Internet Applications (WAI-ARIA) 1.2,” W3C Recommendation, Jun. 2023. [Online]. Available: <https://www.w3.org/TR/wai-aria-1.2/>. [Accessed: Aug. 22, 2026].
- [5] WebAIM, “WAVE Web Accessibility Evaluation Tools.” [Online]. Available: <https://wave.webaim.org/>. [Accessed: Aug. 22, 2026].
- [6] Deque Systems, “Axe DevTools Extension.” [Online]. Available: <https://www.deque.com/axe/devtools/extension/>. [Accessed: Aug. 22, 2026].
- [7] Attention Insight, “AI Attention Analytics and Heatmaps.” [Online]. Available: <https://attentioninsight.com/>. [Accessed: Aug. 22, 2026].
- [8] Microsoft, “TypeScript for the New Programmer,” TypeScript Handbook. [Online]. Available: <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch>. [Accessed: Aug. 22, 2026].
- [9] Microsoft, “TypeScript: Branding.” [Online]. Available: <https://www.typescriptlang.org/branding/>. [Accessed: Aug. 22, 2026].
- [10] OpenJS Foundation, “Don’t Block the Event Loop,” Node.js Documentation. [Online]. Available: <https://nodejs.org/en/learn/asynchronous-work/dont-block-the-event-loop>. [Accessed: Aug. 22, 2026].
- [11] OpenJS Foundation, “Using middleware,” Express Documentation. [Online]. Available: <https://expressjs.com/en/guide/using-middleware/>. [Accessed: Aug. 22, 2026].
- [12] OpenJS Foundation, “Branding of Node.js.” [Online]. Available: <https://nodejs.org/en/about/branding/>. [Accessed: Aug. 22, 2026].
- [13] OpenJS Foundation, “Express.js Logo,” Express.js Community. [Online]. Available: <https://expressjs.com/en/resources/community/>. [Accessed: Aug. 22, 2026].
- [14] Ultralytics, “Object Detection with Ultralytics YOLO.” [Online]. Available: <https://docs.ultralytics.com/tasks/detect/>. [Accessed: Aug. 22, 2026].
- [15] Microsoft, “How to add machine learning to your web application with ONNX Runtime.” [Online]. Available: <https://onnxruntime.ai/docs/tutorials/web/>. [Accessed: Aug. 22, 2026].
- [16] OpenCV, “OpenCV.js Tutorials.” [Online]. Available: https://docs.opencv.org/4.x/d5/d10/tutorial_js_root.html. [Accessed: Aug. 22, 2026].

- [17] Python Software Foundation, “The Python Logo.” [Online]. Available: <https://www.python.org/community/logos/>. [Accessed: Aug. 22, 2026].
- [18] T. Bray, Ed., “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 8259, Dec. 2017, doi: 10.17487/RFC8259.
- [19] Oracle, “MySQL 8.4 Reference Manual.” [Online]. Available: <https://dev.mysql.com/doc/refman/8.4/en/>. [Accessed: Aug. 22, 2026].
- [20] Google, “What is Angular?” Angular Documentation. [Online]. Available: <https://angular.dev/docs>. [Accessed: Aug. 22, 2026].
- [21] Google, “Angular Press Kit.” [Online]. Available: <https://angular.dev/press-kit>. [Accessed: Aug. 22, 2026].
- [22] World Wide Web Consortium, “Media Capture and Streams.” [Online]. Available: <https://www.w3.org/TR/mediacapture-streams/>. [Accessed: Aug. 22, 2026].
- [23] World Wide Web Consortium, “WebXR Device API.” [Online]. Available: <https://www.w3.org/TR/webxr/>. [Accessed: Aug. 22, 2026].
- [24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
- [25] OpenCV, “Official OpenCV Logos & Media Kit.” [Online]. Available: <https://opencv.org/media-kit/>. [Accessed: Aug. 22, 2026].
- [26] Ultralytics, “The Ultralytics Brand.” [Online]. Available: <https://www.ultralytics.com/brand>. [Accessed: Aug. 22, 2026].
- [27] ONNX Runtime developers, “Citing ONNX Runtime.” [Online]. Available: <https://onnxruntime.ai/docs/reference/citing.html>. [Accessed: Aug. 22, 2026].
- [28] Google, “Structured outputs,” Gemini API Documentation. [Online]. Available: <https://ai.google.dev/gemini-api/docs/structured-output>. [Accessed: Aug. 22, 2026].