



**Πρόγραμμα Μεταπτυχιακών Σπουδών
Τμήμα Μηχανικών Πληροφοριακών και
Επικοινωνιακών Συστημάτων**

**Συστήματα Διάχυτου Υπολογισμού
ΕΡΓΑΣΙΑ**

«Smart Door – Face Recognition Access Validation»

Μαυρογιάννη Ελένη

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ: Χρήστος Γκουμόπουλος

ΣΑΜΟΣ 2025



Περιεχόμενα

1. Εισαγωγή	3
2. Πλαίσιο Εφαρμογής και Κίνητρο	5
3. Σχεδιαστική Προσέγγιση και Μεθοδολογία	7
4. Αρχιτεκτονική Συστήματος	9
5. Υλοποίηση	12
6. Οδηγίες Εγκατάστασης και Οδηγός Χρήσης.....	17
7. Σενάρια Χρήσης.....	24
8. Αξιολόγηση και Αποτελέσματα.....	27
9. Περιορισμοί	29
10. Συμπεράσματα και Μελλοντικές Επεκτάσεις.....	31
Αναφορές	33
Παράρτημα Α: Κώδικας	34



Smart Door – Face Recognition Access Validation

1. Εισαγωγή

Ο Διάχυτος Υπολογισμός στοχεύει στη δημιουργία συστημάτων που ενσωματώνονται φυσικά στο περιβάλλον του χρήστη και λειτουργούν συνεχώς ή κατά περίπτωση, αξιοποιώντας δεδομένα περιβάλλοντος, αισθητήρες και μηχανισμούς αυτοματοποιημένης λήψης αποφάσεων. Κεντρική αρχή του πεδίου αποτελεί η προσαρμογή της λειτουργίας ενός συστήματος στο πλαίσιο χρήσης (context), με ελάχιστη απαίτηση άμεσης αλληλεπίδρασης από τον χρήστη, ώστε η τεχνολογία να καθίσταται διακριτική και λειτουργική στην καθημερινότητα.

Στο πλαίσιο της παρούσας εργασίας αναπτύχθηκε το σύστημα Smart Door – Face Recognition Access Validation, ένα έξυπνο σύστημα ελέγχου πρόσβασης που βασίζεται στην αναγνώριση προσώπου. Η βασική λειτουργία του συστήματος είναι η αυτόματη ανίχνευση παρουσίας στην είσοδο ενός χώρου, η ενεργοποίηση της κάμερας, η εκτέλεση αναγνώρισης προσώπου με χρήση τεχνικών υπολογιστικής όρασης μέσω της βιβλιοθήκης OpenCV και η λήψη απόφασης σχετικά με την επιτρεπτότητα της πρόσβασης. Σε περίπτωση επιτυχούς ταυτοποίησης καταγράφεται συμβάν τύπου Access Granted, ενώ σε περίπτωση αποτυχίας ή μη αναγνωρισμένου προσώπου καταγράφεται Access Denied και ενεργοποιείται μηχανισμός ειδοποίησης μέσω ηλεκτρονικού ταχυδρομείου προς τον ιδιοκτήτη του συστήματος.

Η επιλογή του συγκεκριμένου προβλήματος σχετίζεται με την αυξανόμενη ανάγκη για αυτοματοποιημένες, αξιόπιστες και φιλικές προς τον χρήστη λύσεις ασφάλειας, οι οποίες να λειτουργούν αυτόνομα χωρίς την απαίτηση χειροκίνητων διαδικασιών, όπως η χρήση φυσικών κλειδιών ή κωδικών πρόσβασης. Παράλληλα, η προτεινόμενη υλοποίηση αξιοποιεί πολλαπλά υποσυστήματα, όπως λήψη και επεξεργασία εικόνας, ανίχνευση κίνησης σε επίπεδο λογισμικού, web διεπαφή χρήστη, αποθήκευση δεδομένων σε βάση και αποστολή ειδοποιήσεων, καλύπτοντας τις απαιτήσεις ενός ολοκληρωμένου έργου διάχυτου υπολογισμού. Στόχος της εργασίας είναι η σχεδίαση και υλοποίηση ενός ρεαλιστικού και λειτουργικού συστήματος ελέγχου πρόσβασης, καθώς και η τεκμηρίωση των



σχεδιαστικών επιλογών που πραγματοποιήθηκαν. Επιπλέον, αξιολογούνται βασικές μετρικές απόδοσης του συστήματος, όπως ο χρόνος απόκρισης (latency) και η ακρίβεια αναγνώρισης (accuracy), μέσα από πειραματικές δοκιμές σε ελεγχόμενο περιβάλλον.



2. Πλαίσιο Εφαρμογής και Κίνητρο

Τα συστήματα ελέγχου πρόσβασης αποτελούν κρίσιμο στοιχείο της σύγχρονης ασφάλειας σε ιδιωτικούς, επαγγελματικούς και ημιδημόσιους χώρους. Παραδοσιακές μέθοδοι, όπως τα φυσικά κλειδιά, οι κάρτες πρόσβασης και οι κωδικοί PIN, χρησιμοποιούνται ευρέως, ωστόσο παρουσιάζουν σημαντικούς περιορισμούς. Η απώλεια ή η αντιγραφή ενός κλειδιού, η κοινοποίηση ενός κωδικού ή η κλοπή μιας κάρτας μπορούν να οδηγήσουν σε μη εξουσιοδοτημένη πρόσβαση, ενώ παράλληλα απαιτούν ανθρώπινη παρέμβαση για τη διαχείριση, την αντικατάσταση και τη συντήρησή τους.

Η αναγνώριση προσώπου προσφέρει μια πιο σύγχρονη και φυσική προσέγγιση στον έλεγχο πρόσβασης, καθώς βασίζεται σε μοναδικά βιομετρικά χαρακτηριστικά του χρήστη. Με τον τρόπο αυτό, η διαδικασία ταυτοποίησης πραγματοποιείται χωρίς φυσική αλληλεπίδραση, αυξάνοντας τόσο την ευχρηστία όσο και το επίπεδο ασφάλειας. Επιπλέον, η πρόσβαση γίνεται ταχύτερη και πιο φιλική προς τον χρήστη, καθώς δεν απαιτείται απομνημόνευση κωδικών ή μεταφορά φυσικών αντικειμένων.

Στο πλαίσιο του συστήματος Smart Door, η αναγνώριση προσώπου συνδυάζεται με μηχανισμό ανίχνευσης κίνησης, ο οποίος λειτουργεί ως μηχανισμός ενεργοποίησης της διαδικασίας αναγνώρισης. Η ανίχνευση κίνησης υλοποιείται σε επίπεδο λογισμικού, αξιοποιώντας τη ροή εικόνας της κάμερας και τεχνικές επεξεργασίας εικόνας μέσω της βιβλιοθήκης OpenCV. Με τον τρόπο αυτό, η κάμερα και ο αλγόριθμος αναγνώρισης ενεργοποιούνται μόνο όταν ανιχνευθεί πραγματική παρουσία μπροστά στην πόρτα.

Η προσέγγιση αυτή συμβάλλει στη μείωση της άσκοπης επεξεργασίας δεδομένων και της κατανάλωσης υπολογιστικών πόρων, ενώ ταυτόχρονα προσεγγίζει ρεαλιστικά ένα πραγματικό σενάριο χρήσης, στο οποίο το σύστημα λειτουργεί στο παρασκήνιο και παρεμβαίνει μόνο όταν απαιτείται. Παρότι η ανίχνευση κίνησης μέσω κάμερας έχει υψηλότερο υπολογιστικό κόστος σε σύγκριση με έναν φυσικό αισθητήρα, επιλέχθηκε για λόγους απλότητας υλικού και ευελιξίας υλοποίησης.

Το Smart Door σύστημα εντάσσεται στο πλαίσιο του Διάχυτου Υπολογισμού, καθώς ενσωματώνεται φυσικά στο περιβάλλον του χρήστη και λειτουργεί χωρίς συνεχή ανθρώπινη παρέμβαση. Συγκεκριμένα, το σύστημα:



- λειτουργεί αυτόνομα, ενεργοποιούμενο μόνο όταν ανιχνευθεί σχετικό γεγονός,
- αντιλαμβάνεται το περιβάλλον μέσω ανίχνευσης κίνησης και οπτικής πληροφορίας,
- και λαμβάνει αποφάσεις σε πραγματικό χρόνο, χωρίς άμεση εντολή από τον χρήστη.

Η συνολική ιδέα του έργου βασίζεται στη δημιουργία ενός έξυπνου και διακριτικού μηχανισμού ασφάλειας, ο οποίος αξιοποιεί σύγχρονες τεχνικές υπολογιστικής όρασης και λογισμικού, προσφέροντας πρακτική λειτουργικότητα και αποτελώντας μια ρεαλιστική εφαρμογή των αρχών του διάχυτου υπολογισμού.



3. Σχεδιαστική Προσέγγιση και Μεθοδολογία

Ο σχεδιασμός του συστήματος Smart Door βασίστηκε σε μια επεκτάσιμη προσέγγιση, με στόχο τη δημιουργία ενός αυτόνομου συστήματος ελέγχου πρόσβασης που να λειτουργεί αξιόπιστα σε ρεαλιστικές συνθήκες χρήσης. Κατά τον σχεδιασμό δόθηκε έμφαση στη σαφή διάκριση των επιμέρους υποσυστημάτων, ώστε κάθε λειτουργική ενότητα να επιτελεί έναν συγκεκριμένο ρόλο και να μπορεί να τροποποιηθεί ή να επεκταθεί ανεξάρτητα.

Η μεθοδολογία ανάπτυξης ακολούθησε επαναληπτική διαδικασία, κατά την οποία το σύστημα υλοποιήθηκε σταδιακά. Αρχικά αναπτύχθηκαν και ελέγχθηκαν ανεξάρτητα οι βασικές λειτουργίες, όπως η λήψη εικόνας από την κάμερα και ο εντοπισμός προσώπου. Στη συνέχεια, ενσωματώθηκε ο μηχανισμός ανίχνευσης κίνησης σε επίπεδο λογισμικού, ώστε το σύστημα να ενεργοποιείται μόνο όταν ανιχνεύεται σχετικό γεγονός στο περιβάλλον.

Κατά τον σχεδιασμό της ροής λειτουργίας, το σύστημα αντιμετωπίστηκε ως ένα context-aware σύστημα, το οποίο παραμένει σε κατάσταση αναμονής και μεταβαίνει σε ενεργή κατάσταση μόνο όταν το περιβάλλον το απαιτεί. Η ανίχνευση κίνησης λειτουργεί ως γεγονός ενεργοποίησης (trigger), το οποίο οδηγεί στη λήψη εικόνας και στην εκτέλεση της αναγνώρισης προσώπου. Με αυτόν τον τρόπο αποφεύγεται η συνεχής επεξεργασία δεδομένων και επιτυγχάνεται πιο αποδοτική χρήση των διαθέσιμων πόρων.

Ιδιαίτερη σημασία δόθηκε στη σχεδίαση της λήψης αποφάσεων του συστήματος. Η απόφαση για την επιτρεπτότητα της πρόσβασης βασίζεται στο αποτέλεσμα της αναγνώρισης προσώπου και σε προκαθορισμένο κατώφλι εμπιστοσύνης (threshold). Το αποτέλεσμα της απόφασης δεν περιορίζεται μόνο σε οπτική αναπαράσταση, αλλά συνοδεύεται από καταγραφή συμβάντος στη βάση δεδομένων και, σε συγκεκριμένες περιπτώσεις, από ενεργοποίηση μηχανισμού ειδοποίησης.

Η επιλογή τοπικής επεξεργασίας όλων των δεδομένων στο Raspberry Pi αποτέλεσε συνειδητή σχεδιαστική απόφαση. Με αυτόν τον τρόπο διασφαλίζεται η ιδιωτικότητα των δεδομένων, μειώνεται η εξάρτηση από εξωτερικές υπηρεσίες και επιτυγχάνεται λειτουργία ακόμη και σε περιβάλλοντα χωρίς μόνιμη σύνδεση στο διαδίκτυο. Η μόνη εξαίρεση αποτελεί η αποστολή ειδοποιήσεων, η οποία απαιτεί δικτυακή σύνδεση.



Συνολικά, η σχεδιαστική προσέγγιση και η μεθοδολογία που ακολουθήθηκε αποσκοπούν στη δημιουργία ενός απλού αλλά λειτουργικού συστήματος, το οποίο ενσωματώνει τις βασικές αρχές του διάχυτου υπολογισμού και μπορεί να αποτελέσει βάση για περαιτέρω εξέλιξη και εμπλουτισμό με επιπλέον λειτουργίες.



4. Αρχιτεκτονική Συστήματος

Κεντρικό στοιχείο της αρχιτεκτονικής αποτελεί το Raspberry Pi, το οποίο λειτουργεί ως κόμβος επεξεργασίας και συντονισμού. Όλα τα υποσυστήματα εκτελούνται τοπικά στη συσκευή και επικοινωνούν μεταξύ τους μέσω σαφώς καθορισμένων ροών δεδομένων και γεγονότων, όπως η ανίχνευση κίνησης, η λήψη εικόνας και η απόφαση πρόσβασης.

Η συνολική λειτουργία του συστήματος βασίζεται σε γεγονότα (event-driven architecture), με κύριο γεγονός ενεργοποίησης την ανίχνευση κίνησης μπροστά από την κάμερα.

4.1 Υποσύστημα Ανίχνευσης Κίνησης

Το υποσύστημα ανίχνευσης κίνησης στο Smart Door υλοποιήθηκε εξ ολοκλήρου σε επίπεδο λογισμικού, χωρίς τη χρήση φυσικού αισθητήρα PIR. Αντί για ξεχωριστό αισθητήρα, αξιοποιήθηκε η USB κάμερα του συστήματος σε συνδυασμό με τεχνικές επεξεργασίας εικόνας μέσω της βιβλιοθήκης OpenCV.

Η ανίχνευση κίνησης πραγματοποιείται με τη μέθοδο της σύγκρισης διαδοχικών καρέ (frame differencing). Συγκεκριμένα, το σύστημα συγκρίνει διαδοχικά frames από την κάμερα και υπολογίζει τη διαφορά τους. Όταν η διαφορά υπερβεί ένα προκαθορισμένο κατώφλι (area threshold), θεωρείται ότι υπάρχει κίνηση στον χώρο μπροστά από την πόρτα.

Μόνο σε αυτή την περίπτωση ενεργοποιείται η διαδικασία αναγνώρισης προσώπου. Με αυτόν τον τρόπο, αποφεύγεται η συνεχής εκτέλεση του αλγορίθμου αναγνώρισης, μειώνοντας τον υπολογιστικό φόρτο και τη σπατάλη πόρων.

Η συγκεκριμένη προσέγγιση επιλέχθηκε για λόγους απλότητας και ευελιξίας, καθώς:

- δεν απαιτείται πρόσθετος εξοπλισμός,
- μειώνεται η πολυπλοκότητα της φυσικής εγκατάστασης,
- και το σύστημα παραμένει πλήρως λειτουργικό βασιζόμενο αποκλειστικά στην κάμερα.



Παρότι η ανίχνευση κίνησης μέσω κάμερας παρουσιάζει μεγαλύτερο υπολογιστικό κόστος σε σύγκριση με έναν φυσικό αισθητήρα PIR, κρίθηκε επαρκής και κατάλληλη για το εύρος και τον σκοπό της παρούσας εργασίας.

4.2 Υποσύστημα Λήψης Εικόνας

Το υποσύστημα λήψης εικόνας περιλαμβάνει την USB κάμερα που είναι συνδεδεμένη στο Raspberry Pi. Η κάμερα λειτουργεί συνεχώς σε χαμηλό ρυθμό λήψης καρέ για τις ανάγκες της ανίχνευσης κίνησης, ενώ η πλήρης επεξεργασία εικόνας ενεργοποιείται μόνο όταν ανιχνευθεί σχετικό γεγονός.

Μετά την ανίχνευση κίνησης, το σύστημα λαμβάνει το τρέχον καρέ της κάμερας και το προωθεί στο υποσύστημα επεξεργασίας και αναγνώρισης προσώπου. Η λήψη επικεντρώνεται στο κυρίαρχο πρόσωπο του καρέ, σύμφωνα με τους περιορισμούς που έχουν τεθεί στο πλαίσιο της εργασίας.

Η συγκεκριμένη σχεδίαση συμβάλλει:

- στη μείωση του υπολογιστικού φόρτου,
- στην αποφυγή άσκοπης επεξεργασίας δεδομένων,
- και στη βελτίωση της συνολικής απόκρισης του συστήματος.

4.3 Υποσύστημα Επεξεργασίας και Αναγνώρισης

Το υποσύστημα επεξεργασίας και αναγνώρισης υλοποιείται εξ ολοκλήρου στο Raspberry Pi, το οποίο λειτουργεί ως κεντρική μονάδα επεξεργασίας. Για την ανάλυση των εικόνων χρησιμοποιείται η βιβλιοθήκη OpenCV, η οποία παρέχει έτοιμες και αποδοτικές μεθόδους υπολογιστικής όρασης.

Η διαδικασία αναγνώρισης περιλαμβάνει τα ακόλουθα στάδια:

- εντοπισμό προσώπου στο καρέ,
- εξαγωγή χαρακτηριστικών προσώπου,
- σύγκριση των χαρακτηριστικών με πρότυπα αποθηκευμένα στη βάση δεδομένων.

Με βάση το αποτέλεσμα της σύγκρισης και το προκαθορισμένο κατώφλι εμπιστοσύνης (threshold), το σύστημα λαμβάνει απόφαση για την κατάσταση πρόσβασης. Αν το πρόσωπο αντιστοιχεί σε εξουσιοδοτημένο χρήστη, η πρόσβαση



χαρακτηρίζεται ως επιτρεπτή (Access Granted). Σε διαφορετική περίπτωση, η πρόσβαση απορρίπτεται (Access Denied) και ενεργοποιούνται οι αντίστοιχοι μηχανισμοί ειδοποίησης.

4.4 Υποσύστημα Παρουσίας και Ειδοποιήσεων

Το υποσύστημα παρουσίας βασίζεται σε web interface υλοποιημένο με τη χρήση του Flask framework. Το περιβάλλον αυτό παρέχει στον χρήστη μια απλή και κατανοητή εικόνα της κατάστασης του συστήματος, χωρίς σύνθετα γραφικά ή πολύπλοκες αλληλεπιδράσεις.

Μέσω του web interface προβάλλονται:

- η τρέχουσα κατάσταση του συστήματος,
- το αποτέλεσμα της τελευταίας αναγνώρισης,
- και το ιστορικό συμβάντων με χρονική σήμανση.

Παράλληλα, σε περίπτωση μη εξουσιοδοτημένης πρόσβασης, ενεργοποιείται ο μηχανισμός αποστολής ειδοποίησης μέσω email. Ο μηχανισμός αυτός λειτουργεί συμπληρωματικά προς το web interface και επιτρέπει την άμεση ενημέρωση του ιδιοκτήτη, ακόμη και όταν δεν βρίσκεται κοντά στο σύστημα.



5. Υλοποίηση

Το λειτουργικό σύστημα βασίζεται σε Linux και η εφαρμογή αποθηκεύεται σε κάρτα microSD χωρητικότητας 256GB, παρέχοντας επαρκή χώρο για το λειτουργικό σύστημα, τα αρχεία της εφαρμογής, τα δεδομένα εκπαίδευσης προσώπων και τη βάση δεδομένων συμβάντων.

Η ανάπτυξη πραγματοποιήθηκε με στόχο τη ρεαλιστική προσομοίωση ενός έξυπνου συστήματος ελέγχου πρόσβασης, το οποίο μπορεί να λειτουργήσει αυτόνομα και να ενσωματωθεί σε πραγματικό οικιακό περιβάλλον.

5.1 Υλοποίηση Hardware

Το hardware του συστήματος αποτελείται από τα ακόλουθα βασικά στοιχεία:

- **Raspberry Pi 4**

Χρησιμοποιείται ως κεντρική μονάδα επεξεργασίας. Εκτελεί τον αλγόριθμο αναγνώρισης προσώπου, διαχειρίζεται τη βάση δεδομένων και φιλοξενεί το web interface μέσω Flask.

- **USB Camera**

Χρησιμοποιείται για τη λήψη εικόνων του επισκέπτη. Η κάμερα αξιοποιείται τόσο για την ανίχνευση κίνησης σε επίπεδο λογισμικού όσο και για τη λήψη εικόνων που χρησιμοποιούνται στη διαδικασία αναγνώρισης προσώπου.

- **Κάρτα microSD 256GB**

Χρησιμοποιείται για την αποθήκευση του λειτουργικού συστήματος, της εφαρμογής, των αρχείων εκπαίδευσης προσώπων και της βάσης δεδομένων.

Κατά τη φάση ανάπτυξης και δοκιμών, το σύστημα υλοποιήθηκε και επαληθεύτηκε λειτουργικά και με χρήση κάμερας laptop, ώστε να είναι δυνατή η πλήρης δοκιμή του λογισμικού ακόμη και πριν την τελική φυσική εγκατάσταση στο Raspberry Pi.

5.2 Υλοποίηση Software



Η υλοποίηση του λογισμικού βασίστηκε σε ευρέως διαδεδομένες και αξιόπιστες τεχνολογίες:

- **Python**

Η Python χρησιμοποιήθηκε ως κύρια γλώσσα προγραμματισμού, λόγω της ευκολίας ανάπτυξης, της πληθώρας διαθέσιμων βιβλιοθηκών και της πλήρους συμβατότητας με το Raspberry Pi.

- **OpenCV**

Η βιβλιοθήκη OpenCV χρησιμοποιήθηκε για την ανίχνευση κίνησης, τον εντοπισμό προσώπου και την αναγνώριση προσώπων. Η αναγνώριση βασίζεται στην εξαγωγή χαρακτηριστικών και στη σύγκριση με αποθηκευμένα πρότυπα.

- **Flask**

Το Flask χρησιμοποιήθηκε για την υλοποίηση ενός web interface, μέσω του οποίου ο χρήστης μπορεί να παρακολουθεί την κατάσταση του συστήματος και το ιστορικό συμβάντων.

- **SQLite**

Η βάση δεδομένων SQLite χρησιμοποιείται για την αποθήκευση των καταγεγραμμένων γεγονότων, όπως ημερομηνία, ώρα, αποτέλεσμα αναγνώρισης και όνομα προσώπου, όταν αυτό είναι διαθέσιμο.

- **SMTP (Email)**

Ο μηχανισμός SMTP χρησιμοποιείται για την αποστολή ειδοποιήσεων στον ιδιοκτήτη του συστήματος σε περίπτωση εξουσιοδοτημένης πρόσβασης.



5.3 Περιγραφή Λειτουργίας Συστήματος

Η λειτουργία του συστήματος βασίζεται σε μια αυτοματοποιημένη ακολουθία γεγονότων:

1. Η κάμερα παρακολουθεί τον χώρο και το λογισμικό ανιχνεύει κίνηση μέσω σύγκρισης διαδοχικών καρέ.
2. Όταν ανιχνευθεί κίνηση που υπερβαίνει το προκαθορισμένο κατώφλι, ενεργοποιείται η διαδικασία αναγνώρισης.
3. Λαμβάνεται εικόνα του επισκέπτη από την κάμερα.
4. Πραγματοποιείται εντοπισμός προσώπου μέσω OpenCV.
5. Το εντοπισμένο πρόσωπο συγκρίνεται με τα αποθηκευμένα δεδομένα εξουσιοδοτημένων χρηστών.
6. Το σύστημα λαμβάνει απόφαση πρόσβασης, είτε Access Granted είτε Access Denied.
7. Το αποτέλεσμα καταγράφεται στη βάση δεδομένων με χρονική σήμανση.
8. Σε περίπτωση μη αναγνωρισμένου προσώπου, αποστέλλεται αυτόματα ειδοποίηση μέσω email στον ιδιοκτήτη.

Η συγκεκριμένη ροή επιτρέπει στο σύστημα να λειτουργεί αυτόνομα, χωρίς συνεχή παρέμβαση του χρήστη, παρέχοντας παράλληλα άμεση ενημέρωση για κρίσιμα γεγονότα.

5.4 Μεθοδολογία Ανάπτυξης

Η ανάπτυξη του Smart Door συστήματος ακολούθησε επαναληπτική και πειραματική μεθοδολογία. Αρχικά υλοποιήθηκαν και δοκιμάστηκαν ανεξάρτητα τα βασικά υποσυστήματα, όπως η λήψη εικόνας και η αναγνώριση προσώπου.

Στη συνέχεια ενσωματώθηκε ο μηχανισμός ανίχνευσης κίνησης μέσω επεξεργασίας εικόνας, επιτρέποντας στο σύστημα να ενεργοποιείται μόνο όταν υπάρχει πραγματική παρουσία. Ακολούθησε η ανάπτυξη του web interface με Flask και η σύνδεσή του με τη βάση δεδομένων.

Τέλος, υλοποιήθηκε ο μηχανισμός ειδοποιήσεων μέσω email και πραγματοποιήθηκαν επαναλαμβανόμενες δοκιμές ολοκληρωμένης λειτουργίας, με διαδοχικές βελτιώσεις έως την τελική σταθερή έκδοση.



5.5 Εκτέλεση Εφαρμογής

Η εκτέλεση του Smart Door συστήματος πραγματοποιείται μέσω του κεντρικού αρχείου `app.py`, το οποίο λειτουργεί ως σημείο εκκίνησης όλων των υποσυστημάτων.

Η διαδικασία εκτέλεσης περιλαμβάνει τα ακόλουθα βήματα:

1. Εκκίνηση του Raspberry Pi και σύνδεση της USB κάμερας.
2. Ενεργοποίηση του Python virtual environment, εφόσον χρησιμοποιείται.
3. Εκτέλεση της εφαρμογής με την εντολή: `python app.py`
4. Εκκίνηση του web server Flask και των background threads για ανίχνευση κίνησης.
5. Πρόσβαση στο web interface μέσω φυλλομετρητή στη διεύθυνση: `http://127.0.0.1:5000`

Μετά την εκκίνηση, το σύστημα λειτουργεί αυτόνομα και ενεργοποιεί αυτόματα την αναγνώριση προσώπου όταν ανιχνευθεί κίνηση.

5.6 Δομή και Περιγραφή Αρχείων

Η υλοποίηση του συστήματος οργανώνεται σε επιμέρους αρχεία και φακέλους:

- **app.py**

Κεντρικό αρχείο της εφαρμογής. Διαχειρίζεται τη ροή λειτουργίας, τα background threads, το web interface και τον συντονισμό των υποσυστημάτων.

- **config.py**

Περιέχει τις βασικές ρυθμίσεις του συστήματος, όπως paths αρχείων, παραμέτρους κάμερας και ρυθμίσεις ειδοποιήσεων.

- **services/camera.py**

Υλοποιεί τη διασύνδεση με την κάμερα και τη λήψη καρέ.

- **services/MotionDetector.py**

Περιλαμβάνει τον αλγόριθμο ανίχνευσης κίνησης μέσω σύγκρισης διαδοχικών καρέ.

- **services/face_recognizer.py**

Υλοποιεί τον εντοπισμό και την αναγνώριση προσώπου με χρήση OpenCV.



- **services/db.py**

Διαχειρίζεται τη βάση δεδομένων SQLite και την αποθήκευση συμβάντων.

- **services/emailer.py**

Υλοποιεί τον μηχανισμό αποστολής ειδοποιήσεων μέσω email.

- **templates/**

Περιέχει τα αρχεία HTML για το web interface.

Η συγκεκριμένη δομή διασφαλίζει καθαρότητα κώδικα, ευκολία συντήρησης και δυνατότητα μελλοντικής επέκτασης.



6. Οδηγίες Εγκατάστασης και Οδηγός Χρήσης

Η παρούσα ενότητα περιγράφει αναλυτικά τη διαδικασία εγκατάστασης, εκτέλεσης και χρήσης του συστήματος Smart Door – Face Recognition Access Validation. Στόχος είναι η πλήρης καθοδήγηση του χρήστη, από την αρχική προετοιμασία του συστήματος έως την καθημερινή λειτουργία και διαχείρισή του.

6.1 Προαπαιτούμενα

Για τη σωστή λειτουργία του συστήματος απαιτούνται:

- Raspberry Pi 4 με λειτουργικό σύστημα Linux
- USB κάμερα συμβατή με OpenCV
- Εγκατεστημένη Python
- Πρόσβαση στο διαδίκτυο (προαιρετικά, για email ειδοποιήσεις)

6.2 Εγκατάσταση Python και Virtual Environment

Η εφαρμογή υλοποιήθηκε σε Python και χρησιμοποιεί virtual environment για απομόνωση βιβλιοθηκών.

```
sudo apt install python3 python3-pip python3-venv -y
```

```
python3 -m venv venv
```

```
source venv/bin/activate
```

6.3 Εγκατάσταση Απαραίτητων Βιβλιοθηκών

Οι βασικές βιβλιοθήκες εγκαθίστανται μέσω pip, εφόσον υπάρχει αρχείο requirements.txt εκτελούμε:

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

6.4 Σύνδεση και Έλεγχος Κάμερας

Η USB κάμερα συνδέεται στο Raspberry Pi και ελέγχεται η αναγνώρισή της:

```
ls /dev/video*
```



Για δοκιμαστική λήψη εικόνας μπορεί να χρησιμοποιηθεί εργαλείο.

6.5 Εκκίνηση της Εφαρμογής

Η εκτέλεση του συστήματος γίνεται μέσω του αρχείου app.py:

```
python app.py
```

Με την εκκίνηση:

- ενεργοποιείται ο web server Flask,
- ξεκινούν τα background threads για ανίχνευση κίνησης,
- και το σύστημα τίθεται σε κατάσταση αναμονής

6.6 Πρόσβαση στο Web Interface

Ο χρήστης αποκτά πρόσβαση στο σύστημα μέσω φυλλομετρητή:

<http://127.0.0.1:5000>

Μέσω του web interface προβάλλονται:

- η κατάσταση του συστήματος,
- το αποτέλεσμα της τελευταίας αναγνώρισης,
- και το ιστορικό συμβάντων.

The screenshot shows the Smart Door web interface in a browser window. The interface has a light blue header with the title 'Smart Door' and links for 'Email' and 'Settings'. The main content is divided into three sections:

- STATUS:** Shows 'N/A' for NAME, CONFIDENCE, and NOTE. There is a 'Run Recognition' button.
- ADD / TRAIN PERSON:** Includes a 'Person name' input field, a '20' input field, a 'Capture Samples' button, and a 'Train Model' button.
- AUTHORIZED PERSONS:** Shows a list of authorized persons, with 'ellen' visible and a 'Delete' button next to it.

The **RECENT EVENTS** section displays a table of recent face recognition events:

TIME	NAME	RESULT	CONF
2026-01-30T22:41:17	-	NO_FACE	-
2026-01-30T22:41:14	ellen	GRANTED	45.4416004931597
2026-01-30T22:41:12	ellen	GRANTED	49.453190737235836
2026-01-30T22:41:10	-	DENIED	55.02568216584221
2026-01-30T22:41:07	ellen	GRANTED	39.11845668835727
2026-01-30T22:41:05	-	NO_FACE	-
2026-01-30T22:41:02	-	DENIED	57.7092373904058
2026-01-30T22:41:00	-	DENIED	57.10130816621887
2026-01-30T22:40:58	-	DENIED	55.52268781545332
2026-01-30T22:40:53	-	DENIED	54.7873793619505

Εικόνα 1: Παρουσίαση αποτελεσμάτων αναγνώρισης προσώπου και πρόσφατων συμβάντων μέσω του web interface



6.7 Ρύθμιση Email Ειδοποιήσεων

Κατά την πρώτη χρήση, ο χρήστης καλείται να δηλώσει ένα email ειδοποιήσεων.

- Μέσω της σελίδας Email Login, ο χρήστης εισάγει τη διεύθυνση email στην οποία θα αποστέλλονται ειδοποιήσεις.
- Το email αποθηκεύεται στη βάση δεδομένων και χρησιμοποιείται για την αποστολή ειδοποιήσεων σε περίπτωση μη εξουσιοδοτημένης πρόσβασης.

Εικόνα 2: Οθόνη Διαχείρισης Email Ειδοποιήσεων



Smart Door
Email Notifications

ellenmavrogianni@gmail.com

Save email

This email will receive Access Denied alerts and optionally Access Granted events.

Current email:
ellenmavrogianni@gmail.com

Logout and remove email

Εικόνα 3: Διαχείριση Email Ειδοποιήσεων και Ρυθμίσεις Χρήστη

6.8 Εκπαίδευση Χρηστών (Training)

Για να αναγνωρίζει το σύστημα εξουσιοδοτημένα πρόσωπα, απαιτείται αρχικά η εκπαίδευσή του.

Διαδικασία:

1. Ο χρήστης εισάγει το όνομα του προσώπου.
2. Επιλέγει τον αριθμό δειγμάτων (προτεινόμενα 15–30).
3. Πατά το κουμπί Capture Samples, ώστε να καταγραφούν εικόνες προσώπου.
4. Μετά την ολοκλήρωση της καταγραφής, επιλέγει Train Model για την εκπαίδευση του συστήματος.

Η διαδικασία αυτή μπορεί να επαναληφθεί για πολλούς χρήστες.



ADD / TRAIN PERSON

Person name

20

Capture Samples

Train Model

Εικόνα 4: Προσθήκη Νέου Χρήστη και Εκπαίδευση Μοντέλου Αναγνώρισης Προσώπου

6.9 Αυτόματη Λειτουργία Αναγνώρισης

Μετά την εκκίνηση του συστήματος, η λειτουργία είναι πλήρως αυτόματη:

1. Η κάμερα παρακολουθεί τον χώρο σε κατάσταση αναμονής.
2. Όταν ανιχνευθεί κίνηση μέσω επεξεργασίας διαδοχικών καρέ, ενεργοποιείται η αναγνώριση προσώπου.
3. Το σύστημα λαμβάνει απόφαση:
 - **Access Granted** για εξουσιοδοτημένο χρήστη,
 - **Access Denied** για άγνωστο ή μη εξουσιοδοτημένο πρόσωπο.
4. Το συμβάν καταγράφεται στη βάση δεδομένων.
5. Σε περίπτωση εξουσιοδοτημένης ή μη πρόσβασης, αποστέλλεται email ειδοποίηση.

Ο χρήστης δεν χρειάζεται να πατήσει κάποιο κουμπί για τη βασική λειτουργία του συστήματος.

6.10 Χειροκίνητη Ενεργοποίηση (Προαιρετικά)

Για λόγους δοκιμών, παρέχεται η δυνατότητα χειροκίνητης ενεργοποίησης της αναγνώρισης μέσω του κουμπιού Trigger Recognition στο web interface.



Η λειτουργία αυτή χρησιμοποιείται κυρίως για έλεγχο και δοκιμές και δεν απαιτείται κατά την κανονική λειτουργία.

6.11 Διαχείριση Δεδομένων

Μέσω του web interface, ο χρήστης μπορεί:

- να διαγράψει εξουσιοδοτημένους χρήστες,
- να επανεκπαιδεύσει το μοντέλο,
- να καθαρίσει το ιστορικό συμβάντων,
- ή να επαναφέρει το μοντέλο αναγνώρισης.

Settings

Recognition threshold

54

Lower usually accepts more matches. Tune based on your tests.

☒ **Enable email notifications**

☒ **Send email also on Access Granted**

☒ **Attach photo to emails**

Email cooldown (seconds)

30

0 means no cooldown. Example: 30 sends max 1 email per 30 seconds.

Save **Back**

Danger zone

These actions permanently delete data.

Reset model **Clear events**

Εικόνα 5: Ρυθμίσεις Συστήματος και Διαχείριση Ειδοποιήσεων



Εικόνα 6: Διαχείριση Εξουσιοδοτημένων Προσώπων

6.12 Τερματισμός Συστήματος

Ο τερματισμός της εφαρμογής πραγματοποιείται με:

- διακοπή της εκτέλεσης στο terminal (Ctrl + C),
- και στη συνέχεια, προαιρετικά, απενεργοποίηση του Raspberry Pi.



7. Σενάρια Χρήσης

Για την αξιολόγηση της λειτουργικότητας του συστήματος Smart Door, εξετάστηκαν αντιπροσωπευτικά σενάρια χρήσης που προσομοιώνουν πραγματικές καταστάσεις καθημερινής χρήσης. Τα σενάρια αυτά καλύπτουν τόσο την περίπτωση εξουσιοδοτημένης πρόσβασης όσο και την περίπτωση μη εξουσιοδοτημένης παρουσίας, εστιάζοντας στη ροή λειτουργίας και στην αυτόνομη συμπεριφορά του συστήματος.

7.1 Σενάριο 1: Εξουσιοδοτημένος Επισκέπτης

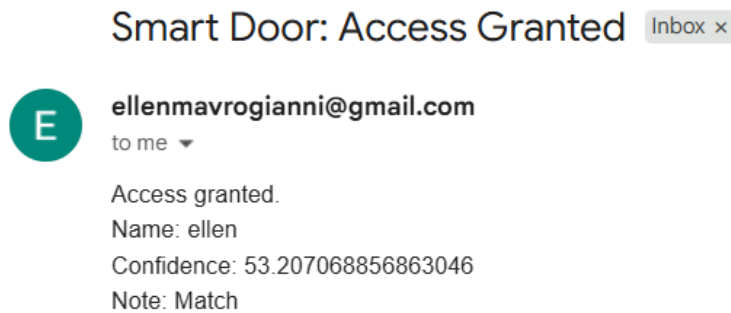
Στο πρώτο σενάριο, ένας εξουσιοδοτημένος χρήστης προσεγγίζει την πόρτα.

Αρχικά, το σύστημα ανιχνεύει κίνηση στον χώρο μπροστά από την είσοδο μέσω της κάμερας, αξιοποιώντας τεχνικές ανίχνευσης κίνησης σε επίπεδο λογισμικού. Η ανίχνευση αυτή ενεργοποιεί αυτόματα τη διαδικασία λήψης εικόνας και αναγνώρισης προσώπου.

Η κάμερα καταγράφει εικόνα του χρήστη και το σύστημα επεξεργάζεται το καρέ, εντοπίζοντας το πρόσωπο του επισκέπτη. Στη συνέχεια, το πρόσωπο συγκρίνεται με τα αποθηκευμένα δεδομένα εξουσιοδοτημένων χρηστών στη βάση δεδομένων. Καθώς υπάρχει επιτυχής αντιστοίχιση, το σύστημα καταλήγει σε απόφαση Access Granted. Το αποτέλεσμα καταγράφεται στη βάση δεδομένων μαζί με τη χρονική σήμανση και το όνομα του χρήστη.

Σε αυτή την περίπτωση μπορεί να αποσταλλεί email ειδοποίηση, εάν το επιθυμεί ο χρήστης. Μετά την ολοκλήρωση της διαδικασίας, το σύστημα επιστρέφει σε κατάσταση αναμονής, έτοιμο να επεξεργαστεί το επόμενο γεγονός.

Το συγκεκριμένο σενάριο επιβεβαιώνει την ορθή λειτουργία του συστήματος σε φυσιολογικές συνθήκες χρήσης και την ικανότητά του να αναγνωρίζει αξιόπιστα εξουσιοδοτημένα πρόσωπα.



Εικόνα 7: Ειδοποίηση Email για Επιτυχή Αναγνώριση Προσώπου (Access Granted)

7.2 Σενάριο 2: Άγνωστος Επισκέπτης

Στο δεύτερο σενάριο, ένας άγνωστος επισκέπτης πλησιάζει την πόρτα.

Όπως και στο προηγούμενο σενάριο, το σύστημα ανιχνεύει κίνηση μέσω της κάμερας και ενεργοποιεί αυτόματα τη διαδικασία λήψης εικόνας και αναγνώρισης προσώπου. Λαμβάνεται εικόνα του επισκέπτη και επιχειρείται ο εντοπισμός και η αναγνώριση του προσώπου.

Καθώς το πρόσωπο δεν αντιστοιχεί σε κάποιο από τα αποθηκευμένα πρότυπα εξουσιοδοτημένων χρηστών, η διαδικασία αναγνώρισης αποτυγχάνει. Το σύστημα λαμβάνει απόφαση Access Denied και καταγράφει το γεγονός στη βάση δεδομένων, μαζί με τις σχετικές πληροφορίες και τη χρονική σήμανση.

Παράλληλα, ενεργοποιείται ο μηχανισμός ειδοποιήσεων και αποστέλλεται αυτόματα email στον ιδιοκτήτη του συστήματος, ενημερώνοντάς τον για την παρουσία μη εξουσιοδοτημένου ατόμου μπροστά στην πόρτα.

Το σενάριο αυτό αναδεικνύει τον ρόλο του συστήματος ως εργαλείο ασφάλειας και επιβεβαιώνει τη δυνατότητα έγκαιρης ειδοποίησης σε περιπτώσεις δυνητικής απειλής, ακόμη και όταν ο χρήστης δεν βρίσκεται στον χώρο.



Smart Door Alert: Access Denied Inbox x



ellenmavrogianni@gmail.com

to me ▼

Access denied.

Name guess: None

Confidence: 54.78737933619505

Note: Unknown or above threshold

Εικόνα 8: Ειδοποίηση Email για Μη Εξουσιοδοτημένη Πρόσβαση (Access Denied)



8. Αξιολόγηση και Αποτελέσματα

Η αξιολόγηση του συστήματος Smart Door πραγματοποιήθηκε μέσω πειραματικών δοκιμών σε ελεγχόμενο περιβάλλον, με σκοπό την εκτίμηση της απόδοσης, της ακρίβειας και της αξιοπιστίας του. Οι δοκιμές επικεντρώθηκαν τόσο στη λειτουργικότητα της αναγνώρισης προσώπου όσο και στη συνολική συμπεριφορά του συστήματος κατά τη συνεχή και αυτόνομη λειτουργία του.

Οι δοκιμές πραγματοποιήθηκαν με τη συμμετοχή 2 έως 5 διαφορετικών ατόμων, τα οποία χρησιμοποιήθηκαν τόσο ως εξουσιοδοτημένοι όσο και ως μη εξουσιοδοτημένοι χρήστες. Ο φωτισμός του χώρου διατηρήθηκε σε ικανοποιητικά επίπεδα, ώστε να προσομοιωθούν ρεαλιστικές συνθήκες λειτουργίας ενός οικιακού περιβάλλοντος.

8.1 Μετρικές Αξιολόγησης

Για την αξιολόγηση του συστήματος χρησιμοποιήθηκαν οι ακόλουθες βασικές μετρικές:

- **Χρόνος απόκρισης:** ο χρόνος από τη στιγμή ανίχνευσης κίνησης μέσω της κάμερας έως την παραγωγή απόφασης Access Granted ή Access Denied.
- **Ακρίβεια αναγνώρισης:** το ποσοστό επιτυχούς αναγνώρισης εξουσιοδοτημένων χρηστών.
- **Αξιοπιστία λειτουργίας:** η σταθερότητα του συστήματος σε επαναλαμβανόμενες δοκιμές και συνεχόμενη λειτουργία.
- **Ορθότητα καταγραφής συμβάντων:** η σωστή αποθήκευση όλων των γεγονότων στη βάση δεδομένων, με πλήρη χρονική σήμανση.

Οι παραπάνω μετρικές επιλέχθηκαν ώστε να καλύπτουν τόσο την τεχνική απόδοση του συστήματος όσο και τη λειτουργική του αξιοπιστία σε συνθήκες πραγματικής χρήσης.

8.2 Αποτελέσματα Πειραμάτων

Τα αποτελέσματα των δοκιμών έδειξαν ότι το σύστημα παρουσιάζει ικανοποιητική απόδοση για το εύρος και τον σκοπό της εφαρμογής:



- Ο χρόνος απόκρισης κυμάνθηκε μεταξύ 0.5 και 1.0 δευτερολέπτων, γεγονός που επιβεβαιώνει ότι το σύστημα λειτουργεί σχεδόν σε πραγματικό χρόνο.
- Η ακρίβεια αναγνώρισης προσώπου κυμάνθηκε μεταξύ 85% και 95%, ανάλογα με τις συνθήκες φωτισμού και τη θέση του χρήστη απέναντι στην κάμερα.
- Το σύστημα παρουσίασε σταθερή λειτουργία σε συνεχή χρήση, χωρίς απροσδόκητες διακοπές ή κρίσιμα σφάλματα.
- Όλα τα συμβάντα (Access Granted, Access Denied, σφάλματα και ειδοποιήσεις) καταγράφηκαν σωστά στη βάση δεδομένων, με πλήρη χρονική σήμανση.

Τα αποτελέσματα αυτά επιβεβαιώνουν ότι το Smart Door σύστημα μπορεί να λειτουργήσει αξιόπιστα σε πραγματικές συνθήκες.

8.3 Παρατηρήσεις και Παράγοντες Επίδρασης

Κατά τη διάρκεια των δοκιμών διαπιστώθηκε ότι η απόδοση του συστήματος επηρεάζεται κυρίως από τους ακόλουθους παράγοντες:

- τον φωτισμό του χώρου, καθώς χαμηλός ή μη ομοιόμορφος φωτισμός δυσκολεύει την ανίχνευση και αναγνώριση προσώπου,
- τη γωνία λήψης της κάμερας, ιδιαίτερα όταν το πρόσωπο δεν βρίσκεται πλήρως στραμμένο προς τον φακό,
- και τη σταθερότητα της εικόνας κατά τη λήψη, ειδικά σε περιπτώσεις απότομης κίνησης.

Παρόλα αυτά, ακόμη και υπό λιγότερο ιδανικές συνθήκες, το σύστημα παρέμεινε λειτουργικό και παρήγαγε συνεπείς αποφάσεις, γεγονός που επιβεβαιώνει την ανθεκτικότητα και τη χρηστικότητά του στο πλαίσιο της εφαρμογής.



9. Περιορισμοί

Παρότι το σύστημα Smart Door επιτυγχάνει τον βασικό του στόχο και λειτουργεί ικανοποιητικά στο πλαίσιο της παρούσας εργασίας, εντοπίζονται ορισμένοι περιορισμοί που σχετίζονται τόσο με την υλοποίηση όσο και με τις συνθήκες λειτουργίας του.

Ένας βασικός περιορισμός αφορά την υποστήριξη ενός μόνο προσώπου ανά καρέ. Το σύστημα έχει σχεδιαστεί ώστε να επεξεργάζεται το κυρίαρχο πρόσωπο που εντοπίζεται από την κάμερα, με βάση το μέγεθος και τη θέση του στο κάδρο. Σε περιπτώσεις όπου εμφανίζονται ταυτόχρονα περισσότερα από ένα άτομα, δεν είναι δυνατή η ανεξάρτητη και ταυτόχρονη αναγνώριση όλων των προσώπων, γεγονός που μπορεί να οδηγήσει σε μερική ή ελλιπή αποτύπωση της πραγματικής κατάστασης.

Κατά τη διάρκεια των δοκιμών παρατηρήθηκε ότι το σύστημα αναγνωρίζει με υψηλή ακρίβεια εξουσιοδοτημένους χρήστες όταν αυτοί βρίσκονται φυσικά μπροστά στην κάμερα, παράγοντας απόφαση *Access Granted*. Ωστόσο, όταν παρουσιάστηκε στην κάμερα στατική φωτογραφία εξουσιοδοτημένου χρήστη, το σύστημα παρήγαγε απόφαση *Access Denied*. Η συμπεριφορά αυτή οφείλεται στο γεγονός ότι ο αλγόριθμος αναγνώρισης βασίζεται σε χαρακτηριστικά που εξάγονται από ζωντανή ροή εικόνας και επηρεάζονται από παράγοντες όπως η φυσική κίνηση, η γωνία λήψης και το βάθος του προσώπου. Οι στατικές εικόνες δεν περιέχουν επαρκή πληροφορία για αξιόπιστη αντιστοίχιση με τα αποθηκευμένα πρότυπα.

Παρότι το σύστημα δεν υλοποιεί ρητό μηχανισμό ελέγχου liveness, η αδυναμία αναγνώρισης προσώπων μέσω φωτογραφίας λειτουργεί έμμεσα ως μηχανισμός αποτροπής απλών επιθέσεων που βασίζονται σε στατικές εικόνες.

Επιπλέον, παρατηρήθηκαν περιορισμοί στη συμπεριφορά του συστήματος σε περιπτώσεις ταυτόχρονης παρουσίας πολλών ατόμων μπροστά στην κάμερα. Όταν στο κάδρο εμφανίζονταν δύο εξουσιοδοτημένοι χρήστες, το σύστημα κατάφερνε συνήθως να αναγνωρίσει τουλάχιστον έναν από αυτούς και να αποδώσει απόφαση *Access Granted*. Ωστόσο, όταν στο ίδιο καρέ βρισκόταν ένας εξουσιοδοτημένος και ένας μη εξουσιοδοτημένος χρήστης, το σύστημα τείνει να αναγνωρίζει μόνο τον εξουσιοδοτημένο, αγνοώντας ή μη επεξεργαζόμενο επαρκώς το δεύτερο



πρόσωπο. Ως αποτέλεσμα, η παρουσία μη εξουσιοδοτημένου χρήστη ενδέχεται να μην οδηγήσει σε απόφαση *Access Denied*, γεγονός που υπογραμμίζει τον περιορισμό της μονοπρόσωπης επεξεργασίας.

Ένας ακόμη σημαντικός περιορισμός αφορά τη στάση και τη γωνία του προσώπου σε σχέση με την κάμερα. Διαπιστώθηκε ότι όταν το πρόσωπο δεν βρίσκεται σε μετωπική θέση ή παρουσιάζει έντονη κλίση, η ακρίβεια της αναγνώρισης μειώνεται αισθητά. Σε τέτοιες περιπτώσεις, το σύστημα ενδέχεται να αποτύχει να αναγνωρίσει ακόμη και εξουσιοδοτημένους χρήστες. Το φαινόμενο αυτό επιβεβαιώθηκε και μέσω δοκιμών με συνεχή ροή βίντεο.

Επιπρόσθετα, οι συνθήκες φωτισμού επηρεάζουν την απόδοση τόσο της ανίχνευσης όσο και της αναγνώρισης προσώπου. Χαμηλός ή μη ομοιόμορφος φωτισμός δυσχεραίνει τον εντοπισμό χαρακτηριστικών του προσώπου, αυξάνοντας την πιθανότητα αποτυχημένων αναγνωρίσεων.

Τέλος, παρατηρήθηκε ότι ο μηχανισμός αποστολής ειδοποιήσεων μέσω email δεν ενεργοποιείται πάντα με απόλυτη συνέπεια σε όλες τις περιπτώσεις μη εξουσιοδοτημένης πρόσβασης. Σε ορισμένες δοκιμές, παρότι καταγράφηκε γεγονός *Access Denied*, δεν πραγματοποιήθηκε επιτυχής αποστολή ειδοποίησης. Ο περιορισμός αυτός σχετίζεται κυρίως με εξωτερικούς παράγοντες, όπως ρυθμίσεις SMTP, περιορισμούς παρόχων email ή ζητήματα δικτυακής συνδεσιμότητας, και όχι με τον βασικό μηχανισμό αναγνώρισης προσώπου.

Οι παραπάνω περιορισμοί δεν αναιρούν τη λειτουργικότητα του συστήματος Smart Door, αλλά αποτυπώνουν με σαφήνεια τα όρια της παρούσας υλοποίησης και αποτελούν αφετηρία για μελλοντικές βελτιώσεις και επεκτάσεις.



10. Συμπεράσματα και Μελλοντικές Επεκτάσεις

Στο πλαίσιο της παρούσας εργασίας υλοποιήθηκε ένα έξυπνο σύστημα ελέγχου πρόσβασης με την ονομασία *Smart Door – Face Recognition Access Validation*, το οποίο εντάσσεται πλήρως στο πεδίο του Διάχυτου Υπολογισμού. Το σύστημα σχεδιάστηκε ώστε να λειτουργεί αυτόνομα, να αντιλαμβάνεται το περιβάλλον μέσω αισθητηριακής πληροφορίας και να λαμβάνει αποφάσεις χωρίς άμεση παρέμβαση του χρήστη, αξιοποιώντας έννοιες όπως το context-awareness και η γεγοντοκεντρική λειτουργία.

Η συνδυαστική χρήση ανίχνευσης κίνησης σε επίπεδο λογισμικού και αναγνώρισης προσώπου επιτρέπει στο σύστημα να ενεργοποιείται μόνο όταν υπάρχει πραγματική παρουσία μπροστά στην κάμερα. Με τον τρόπο αυτό μειώνεται η άσκοπη επεξεργασία δεδομένων, περιορίζεται ο υπολογιστικός φόρτος και ενισχύεται η ρεαλιστικότητα της εφαρμογής, προσεγγίζοντας σενάρια πραγματικής χρήσης σε οικιακό περιβάλλον.

Η αξιολόγηση του συστήματος έδειξε ότι επιτυγχάνει ικανοποιητική ακρίβεια αναγνώρισης και μικρό χρόνο απόκρισης, επιτρέποντας τη λήψη αποφάσεων σχεδόν σε πραγματικό χρόνο. Παράλληλα, η καταγραφή όλων των συμβάντων στη βάση δεδομένων και η δυνατότητα αποστολής ειδοποιήσεων μέσω email ενισχύουν τη χρηστικότητα του συστήματος και την αίσθηση ασφάλειας για τον χρήστη. Η απλή web διεπαφή καθιστά το σύστημα εύχρηστο και προσβάσιμο, χωρίς να απαιτείται εξειδικευμένη τεχνική γνώση.

Παρά τα θετικά αποτελέσματα, η παρούσα υλοποίηση παρουσιάζει ορισμένους περιορισμούς, οι οποίοι αναδεικνύουν ταυτόχρονα και τις δυνατότητες για μελλοντική εξέλιξη.

Μια σημαντική κατεύθυνση μελλοντικής επέκτασης αφορά την ενσωμάτωση ρητών μηχανισμών ελέγχου liveness, ώστε να διασφαλίζεται με σαφή και ελεγχόμενο τρόπο ότι το πρόσωπο που αναγνωρίζεται ανήκει σε πραγματικό άτομο και όχι σε στατική εικόνα ή βίντεο. Στην παρούσα υλοποίηση, αν και δεν υφίσταται μηχανισμός liveness, παρατηρήθηκε ότι στατικές φωτογραφίες εξουσιοδοτημένων χρηστών δεν αναγνωρίζονται επιτυχώς, γεγονός που λειτουργεί έμμεσα ως περιορισμένη αποτροπή απλών επιθέσεων.



Επιπλέον, το σύστημα δεν υποστηρίζει ταυτόχρονη και ανεξάρτητη αναγνώριση πολλαπλών προσώπων στο ίδιο καρέ, καθώς έχει σχεδιαστεί να επεξεργάζεται ένα κυρίαρχο πρόσωπο ανά χρονική στιγμή. Η μελλοντική υποστήριξη πολυπρόσωπης αναγνώρισης θα επέτρεπε την ακριβέστερη αποτύπωση σύνθετων σεναρίων χρήσης και την ασφαλέστερη αξιολόγηση περιπτώσεων όπου συνυπάρχουν εξουσιοδοτημένοι και μη εξουσιοδοτημένοι χρήστες.

Επιπρόσθετα, η ανάπτυξη mobile εφαρμογής θα μπορούσε να προσφέρει πιο άμεση και ευέλικτη διαχείριση του συστήματος, επιτρέποντας στον χρήστη να λαμβάνει ειδοποιήσεις και να παρακολουθεί την κατάσταση της πόρτας απομακρυσμένα. Σε αυτό το πλαίσιο, η αντικατάσταση ή συμπλήρωση των email ειδοποιήσεων με push notifications θα μπορούσε να βελτιώσει περαιτέρω την αμεσότητα της ενημέρωσης.

Τέλος, η μελλοντική αποθήκευση δεδομένων σε υποδομές cloud θα μπορούσε να προσφέρει καλύτερη κλιμάκωση, αυξημένη διαθεσιμότητα και δυνατότητες ανάλυσης ιστορικών δεδομένων, επεκτείνοντας τη χρήση του συστήματος πέρα από ένα μεμονωμένο περιβάλλον.

Συνολικά, το *Smart Door* project αποτελεί μια ρεαλιστική και λειτουργική προσέγγιση ενός συστήματος έξυπνης πρόσβασης, αναδεικνύοντας στην πράξη βασικές αρχές του Διάχυτου Υπολογισμού και αποτελώντας μια σταθερή βάση για περαιτέρω εξέλιξη προς πιο ολοκληρωμένες, αποδοτικές και ασφαλείς λύσεις.



Αναφορές

1. <https://devdocs.io/flask/>
2. <https://docs.opencv.org/4.x/>
3. https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
4. https://itsupport.umd.edu/itsupport?id=kb_article_view&sysparm_article=KB0015112
5. <https://www.raspberrypi.com/documentation/>
6. <https://www.raspberrypi.com/documentation/computers/getting-started.html>
7. <https://www.raspberrypi.com/products/raspberry-pi-400/specifications/>
8. <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>
9. <https://docs.python.org/3/library/smtplib.html>



Παράρτημα Α: Κώδικας

Services/Camera.py

```
import cv2
import os
import time
import threading

class Camera:
    def __init__(self, index: int = 0):
        self.index = index
        self.cap = None
        self.lock = threading.Lock()

    def open(self) -> None:
        if self.cap is not None:
            return

        if os.name == "nt":
            cap = cv2.VideoCapture(self.index, cv2.CAP_DSHOW)
        else:
            cap = cv2.VideoCapture(self.index, cv2.CAP_V4L2)

        if not cap.isOpened():
            cap.release()
            raise RuntimeError(f"Camera open failed
(index={self.index})")

        for _ in range(10):
            cap.read()
            time.sleep(0.03)

        self.cap = cap

    def close(self) -> None:
        with self.lock:
            if self.cap is not None:
                self.cap.release()
                self.cap = None

    def capture_frame(self):
        with self.lock:
            if self.cap is None:
                self.open()

            ok, frame = self.cap.read()
            if not ok or frame is None:
```



```
        raise RuntimeError("Camera frame capture failed")

    return frame.copy()
```

Services/db.py

```
import sqlite3
from typing import Optional, Dict, Any, List
from datetime import datetime

class DB:
    def __init__(self, db_path: str):
        self.db_path = db_path
        self._init_db()

    def _connect(self):
        conn = sqlite3.connect(self.db_path, check_same_thread=False)
        conn.row_factory = sqlite3.Row
        return conn

    def _init_db(self):
        conn = self._connect()
        cur = conn.cursor()

        cur.execute("""
        CREATE TABLE IF NOT EXISTS events (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            ts TEXT NOT NULL,
            name TEXT,
            result TEXT NOT NULL,
            confidence REAL,
            note TEXT
        )
        """)

        cur.execute("""
        CREATE TABLE IF NOT EXISTS settings (
            key TEXT PRIMARY KEY,
            value TEXT NOT NULL
        )
        """)

        conn.commit()
        conn.close()

    def set_setting(self, key: str, value: str):
        conn = self._connect()
        cur = conn.cursor()
        cur.execute("""
```



```
        INSERT INTO settings(key, value) VALUES(?, ?)
        ON CONFLICT(key) DO UPDATE SET value=excluded.value
        """, (key, value))
    conn.commit()
    conn.close()

    def get_setting(self, key: str, default: Optional[str] = None) ->
    str:
        conn = self._connect()
        cur = conn.cursor()
        cur.execute("SELECT value FROM settings WHERE key=?", (key,))
        row = cur.fetchone()
        conn.close()
        return row["value"] if row else (default if default is not None
    else "")

    def add_event(self, name: Optional[str], result: str, confidence:
    Optional[float], note: str = ""):
        conn = self._connect()
        cur = conn.cursor()
        ts = datetime.now().isoformat(timespec="seconds")
        cur.execute(
            "INSERT INTO events(ts, name, result, confidence, note)
VALUES(?,?,?,?,?)",
            (ts, name, result, confidence, note)
        )
        conn.commit()
        conn.close()

    def latest_events(self, limit: int = 30) -> List[Dict[str, Any]]:
        conn = self._connect()
        cur = conn.cursor()
        cur.execute("SELECT * FROM events ORDER BY id DESC LIMIT ?",
(limit,))
        rows = cur.fetchall()
        conn.close()
        return [dict(r) for r in rows]

    def clear_events(self):
        cur = self.conn.cursor()
        cur.execute("DELETE FROM events")
        self.conn.commit()
```

Services/emailer.py

```
import smtplib
from email.message import EmailMessage
from typing import Optional

class EMailer:
```



```
def __init__(self, host, port, username, password, email_from,
email_to):
    self.host = host
    self.port = port
    self.username = username
    self.password = password
    self.email_from = email_from
    self.email_to = email_to

def send(self, subject: str, body: str, image_jpg_bytes:
Optional[bytes] = None):
    msg = EmailMessage()
    msg["Subject"] = subject
    msg["From"] = self.email_from
    msg["To"] = self.email_to
    msg.set_content(body)

    if image_jpg_bytes:
        msg.add_attachment(
            image_jpg_bytes,
            maintype="image",
            subtype="jpeg",
            filename="visitor.jpg"
        )

    with smtplib.SMTP(self.host, self.port, timeout=20) as s:
        s.ehlo()
        s.starttls()
        s.ehlo()
        s.login(self.username, self.password)
        s.send_message(msg)

def send_access_denied(self, body: str, image_jpg_bytes:
Optional[bytes] = None):
    self.send("Smart Door Alert: Access Denied", body,
image_jpg_bytes)

def send_access_granted(self, body: str, image_jpg_bytes:
Optional[bytes] = None):
    self.send("Smart Door: Access Granted", body, image_jpg_bytes)
```

Services/face_recognizer.py

```
import json
import cv2
import numpy as np
from pathlib import Path
from typing import Optional, Tuple
```



```
class FaceRecognizer:
    def __init__(
        self,
        model_path: str,
        labels_path: str,
        threshold: float = 60.0,
        min_labels_required: int = 1,
    ):
        self.model_path = model_path
        self.labels_path = labels_path
        self.threshold = float(threshold)
        self.min_labels_required = int(min_labels_required)

        self.face_cascade = cv2.CascadeClassifier(
            cv2.data.haarcascades +
            "haarcascade_frontalface_default.xml"
        )

        self.recognizer = cv2.face.LBPHFaceRecognizer_create()
        self.labels = {}
        self._load()

    def _load(self):
        if Path(self.labels_path).exists():
            with open(self.labels_path, "r", encoding="utf-8") as f:
                self.labels = json.load(f) or {}
        else:
            self.labels = {}

        if Path(self.model_path).exists():
            self.recognizer.read(self.model_path)

    def set_threshold(self, value: float):
        self.threshold = float(value)

    def detect_and_recognize(self, frame) -> Tuple[str, Optional[str],
Optional[float], str]:

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.equalizeHist(gray)

        faces = self.face_cascade.detectMultiScale(
            gray, scaleFactor=1.2, minNeighbors=5, minSize=(80, 80)
        )

        if len(faces) == 0:
            return "NO_FACE", None, None, "No face detected"

        faces = sorted(faces, key=lambda r: r[2] * r[3], reverse=True)
```



```
(x, y, w, h) = faces[0]
roi = gray[y:y + h, x:x + w]

if len(self.labels) < self.min_labels_required:
    return "DENIED", None, None, f"Insufficient trained persons
({len(self.labels)})."

try:
    label_id, confidence = self.recognizer.predict(roi)
except cv2.error:
    return "DENIED", None, None, "Model not trained yet"

name = self.labels.get(str(label_id), None)

if confidence <= self.threshold and name is not None:
    return "GRANTED", name, float(confidence), "Match"

return "DENIED", None, float(confidence), "Unknown or above
threshold"

@staticmethod
def train_from_folder(faces_dir: str, model_path: str, labels_path:
str) -> Tuple[int, int]:
    faces_path = Path(faces_dir)
    if not faces_path.exists():
        raise RuntimeError("faces_dir does not exist")

    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades +
        "haarcascade_frontalface_default.xml"
    )
    recognizer = cv2.face.LBPHFaceRecognizer_create()

    X = []
    y = []
    labels = {}
    label_to_id = {}
    next_id = 0

    total_imgs = 0
    used_faces = 0

    for person_dir in sorted([p for p in faces_path.iterdir() if
p.is_dir()]):
        person = person_dir.name
        if person not in label_to_id:
            label_to_id[person] = next_id
            labels[str(next_id)] = person
```



```
next_id += 1

for img_path in person_dir.glob("*."):
    total_imgs += 1
    img = cv2.imread(str(img_path))
    if img is None:
        continue

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.2, 5,
minSize=(80, 80))
    if len(faces) == 0:
        continue

    faces = sorted(faces, key=lambda r: r[2] * r[3],
reverse=True)

    x, y0, w, h = faces[0]
    roi = gray[y0:y0 + h, x:x + w]

    X.append(roi)
    y.append(label_to_id[person])
    used_faces += 1

if used_faces < 2:
    raise RuntimeError("Not enough training samples")

labels_np = np.array(y, dtype=np.int32).reshape(-1, 1)
recognizer.train(X, labels_np)

Path(model_path).parent.mkdir(parents=True, exist_ok=True)
Path(labels_path).parent.mkdir(parents=True, exist_ok=True)

recognizer.write(model_path)
with open(labels_path, "w", encoding="utf-8") as f:
    json.dump(labels, f, ensure_ascii=False, indent=2)

return total_imgs, used_faces
```

Services/MotionDetector.py

```
import cv2

class MotionDetector:
    def __init__(self, area_threshold=2500):
        self.prev_gray = None
        self.area_threshold = area_threshold

    def detect(self, frame) -> bool:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```




```
gray = cv2.GaussianBlur(gray, (21, 21), 0)

if self.prev_gray is None:
    self.prev_gray = gray
    return False

diff = cv2.absdiff(self.prev_gray, gray)
thresh = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)[1]
thresh = cv2.dilate(thresh, None, iterations=2)

contours, _ = cv2.findContours(
    thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
)

self.prev_gray = gray

for c in contours:
    if cv2.contourArea(c) > self.area_threshold:
        return True

return False
```

config.py

```
from dataclasses import dataclass
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent

@dataclass
class Config:
    DB_PATH: str = str(BASE_DIR / "data" / "smartdoor.db")
    MODEL_PATH: str = str(BASE_DIR / "data" / "models" / "lbph.yml")
    LABELS_PATH: str = str(BASE_DIR / "data" / "models" /
"labels.json")
    FACES_DIR: str = str(BASE_DIR / "data" / "faces")
    LOG_PATH: str = str(BASE_DIR / "logs" / "smartdoor.log")

    CAMERA_INDEX: int = 0

    PIR_ENABLED: bool = False
    PIR_GPIO_PIN: int = 17

    EMAIL_ENABLED_DEFAULT: bool = True
    EMAIL_TO: str = "youremail@gmail.com"
    EMAIL_FROM: str = "youremail@gmail.com"
    SMTP_HOST: str = "smtp.gmail.com"
    SMTP_PORT: int = 587
    SMTP_USERNAME: str = "youremail@gmail.com"
    SMTP_PASSWORD: str = "yourAppPassword"
```



```
DEFAULT_THRESHOLD: float = 60.0

CONFIG = Config()

App.py
import os
import threading
import time
import shutil
import cv2
from flask import Flask, render_template, redirect, url_for, request,
flash, Response, session

from config import CONFIG
from services.db import DB
from services.camera import Camera
from services.face_recognizer import FaceRecognizer
from services.emailer import Emailer
from services.MotionDetector import MotionDetector

app = Flask(__name__)
app.secret_key = "change-me"

db = DB(CONFIG.DB_PATH)
camera = Camera(CONFIG.CAMERA_INDEX)

# Global motion detector
try:
    motion_detector = MotionDetector(area_threshold=2500)
    warm = camera.capture_frame()
    motion_detector.detect(warm)
    print("Motion detector ready")
except Exception as e:
    print("Motion disabled:", e)
    motion_detector = None

recognizer_lock = threading.Lock()
recognizer = FaceRecognizer(
    CONFIG.MODEL_PATH,
    CONFIG.LABELS_PATH,
    threshold=0.0
)

emailer = Emailer(
    CONFIG.SMTP_HOST,
    CONFIG.SMTP_PORT,
```



```
CONFIG.SMTP_USERNAME,  
CONFIG.SMTP_PASSWORD,  
CONFIG.EMAIL_FROM,  
CONFIG.EMAIL_TO,  
)  
  
status_lock = threading.Lock()  
status = {  
    "camera": "standby",  
    "last_result": "N/A",  
    "last_name": None,  
    "last_confidence": None,  
    "note": "",  
}  
  
last_email_sent_ts = 0.0  
  
def get_threshold() -> float:  
    v = db.get_setting("threshold", str(CONFIG.DEFAULT_THRESHOLD))  
    try:  
        return float(v)  
    except Exception:  
        return CONFIG.DEFAULT_THRESHOLD  
  
def get_notifications_enabled() -> bool:  
    v = db.get_setting("email_enabled", "1" if  
CONFIG.EMAIL_ENABLED_DEFAULT else "0")  
    return v == "1"  
  
def get_email_on_granted() -> bool:  
    v = db.get_setting("email_on_granted", "0")  
    return v == "1"  
  
def get_attach_photo() -> bool:  
    v = db.get_setting("attach_photo", "1")  
    return v == "1"  
  
def get_email_cooldown_sec() -> int:  
    v = db.get_setting("email_cooldown_sec", "30")  
    try:  
        n = int(v)  
        if n < 0:  
            n = 0  
        if n > 3600:
```



```
        n = 3600
    return n
except Exception:
    return 30

def get_notify_email() -> str:
    v = db.get_setting("notify_email", "")
    return (v or "").strip()

def set_notify_email(email: str):
    db.set_setting("notify_email", (email or "").strip())

def _maybe_encode_frame_jpg(frame):
    if not get_attach_photo():
        return None
    try:
        ok, buf = cv2.imencode(".jpg", frame)
        if ok:
            return buf.tobytes()
    except Exception:
        return None
    return None

def _can_send_email(now_ts: float) -> bool:
    global last_email_sent_ts
    cooldown = get_email_cooldown_sec()
    if cooldown == 0:
        return True
    return (now_ts - last_email_sent_ts) >= cooldown

def _mark_email_sent(now_ts: float):
    global last_email_sent_ts
    last_email_sent_ts = now_ts

def _make_emailer_for_current_recipient() -> Emitter:
    to_email = get_notify_email() or CONFIG.EMAIL_TO
    from_email = getattr(CONFIG, "EMAIL_FROM", CONFIG.SMTP_USERNAME)
    return Emitter(
        CONFIG.SMTP_HOST,
        CONFIG.SMTP_PORT,
        CONFIG.SMTP_USERNAME,
        CONFIG.SMTP_PASSWORD,
        from_email,
        to_email,
```



```
)

def process_one_attempt(source: str = "manual", frame=None):
    global status

    with status_lock:
        status["camera"] = "capturing"
        status["note"] = f"Triggered by {source}"

    try:
        if frame is None:
            frame = camera.capture_frame()

        with recognizer_lock:
            recognizer.set_threshold(get_threshold())
            result, name, confidence, note =
recognizer.detect_and_recognize(frame)

            if result == "GRANTED":
                db.add_event(name=name, result="GRANTED",
confidence=confidence, note=note)
            elif result == "DENIED":
                db.add_event(name=name, result="DENIED",
confidence=confidence, note=note)
            else:
                db.add_event(name=None, result="NO_FACE", confidence=None,
note=note)

            if get_notifications_enabled():
                if not get_notify_email():
                    db.add_event(name=name, result="EMAIL_SKIP",
confidence=confidence, note="No notify_email set")
                else:
                    now = time.time()
                    if _can_send_email(now):
                        image_bytes = _maybe_encode_frame_jpg(frame)
                        try:
                            emailer_local =
_make_emailer_for_current_recipient()

                            if result == "DENIED":
                                emailer_local.send_access_denied(
                                    f"Access denied.\nName guess:
{name}\nConfidence: {confidence}\nNote: {note}",
                                    image_jpg_bytes=image_bytes,
                                )
                                _mark_email_sent(now)
```



```
elif result == "GRANTED" and
get_email_on_granted():
    emailer_local.send_access_granted(
        f"Access granted.\nName:
{name}\nConfidence: {confidence}\nNote: {note}",
        image_jpg_bytes=image_bytes,
    )
    _mark_email_sent(now)

except Exception as e:
    db.add_event(name=name, result="EMAIL_FAIL",
confidence=confidence, note=str(e))

    with status_lock:
        status["last_result"] = result
        status["last_name"] = name
        status["last_confidence"] = confidence
        status["note"] = note

except Exception as e:
    db.add_event(name=None, result="ERROR", confidence=None,
note=str(e))
    with status_lock:
        status["last_result"] = "ERROR"
        status["note"] = str(e)
finally:
    with status_lock:
        status["camera"] = "standby"

def motion_loop():
    global motion_detector

    print("Motion detection loop started")
    if motion_detector is None:
        print("Motion detector disabled")
        return

    while True:
        try:
            frame = camera.capture_frame()

            if motion_detector.detect(frame):
                process_one_attempt(source="MOTION", frame=frame)
                time.sleep(2.0)
            else:
                time.sleep(0.1)

        except Exception as e:
```



```
        db.add_event(name=None, result="MOTION_ERROR",
confidence=None, note=str(e))
        time.sleep(1.0)

def capture_samples(person_name: str, num_samples: int):
    person_name = person_name.strip()
    if not person_name:
        raise ValueError("Person name required")

    save_dir = os.path.join(CONFIG.FACES_DIR, person_name)
    os.makedirs(save_dir, exist_ok=True)

    face_cascade = cv2.CascadeClassifier(
        cv2.data.harcascades + "haarcascade_frontalface_default.xml"
    )

    saved = 0
    attempts = 0

    while saved < num_samples and attempts < num_samples * 8:
        attempts += 1
        frame = camera.capture_frame()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.2, 5,
minSize=(80, 80))
        if len(faces) == 0:
            time.sleep(0.08)
            continue
        faces = sorted(faces, key=lambda r: r[2] * r[3], reverse=True)
        x, y, w, h = faces[0]
        roi = gray[y:y + h, x:x + w]
        cv2.imwrite(os.path.join(save_dir,
f"{int(time.time()*1000)}_{saved}.jpg"), roi)
        saved += 1
        time.sleep(0.08)

    if saved == 0:
        raise RuntimeError("No face samples captured")

    return saved

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form.get("email", "").strip()

        if not email or "@" not in email or "." not in email:
```



```
flash("Please enter a valid email address.")
return redirect(url_for("login"))

set_notify_email(email)
session["configured"] = True
flash("Done! Notifications will be sent to this email
address.")
return redirect(url_for("index"))

current_email = get_notify_email()
return render_template("login.html", current_email=current_email)

@app.route("/logout", methods=["POST"])
def logout():
    set_notify_email("")
    session.pop("configured", None)
    flash("The notification email has been cleared.")
    return redirect(url_for("login"))

@app.route("/")
def index():
    if not get_notify_email():
        return redirect(url_for("login"))

    events = db.latest_events(30)
    with status_lock:
        st = dict(status)

    persons = []
    if os.path.isdir(CONFIG.FACES_DIR):
        persons = sorted(
            d for d in os.listdir(CONFIG.FACES_DIR)
            if os.path.isdir(os.path.join(CONFIG.FACES_DIR, d))
        )

    return render_template("index.html", status=st, events=events,
persons=persons)

@app.route("/trigger", methods=["POST"])
def trigger():
    threading.Thread(target=process_one_attempt, args=("MANUAL",),
daemon=True).start()
    flash("Started recognition attempt.")
    return redirect(url_for("index"))

@app.route("/status_json")
```




```
def status_json():
    with status_lock:
        return {
            "status": status,
            "events": db.latest_events(10)
        }

@app.route("/train", methods=["POST"])
def train():
    global recognizer
    try:
        with recognizer_lock:
            total, used = FaceRecognizer.train_from_folder(
                CONFIG.FACES_DIR,
                CONFIG.MODEL_PATH,
                CONFIG.LABELS_PATH
            )
            recognizer = FaceRecognizer(
                CONFIG.MODEL_PATH,
                CONFIG.LABELS_PATH,
                threshold=get_threshold()
            )
            flash(f"Training complete. Images: {total}, Faces used: {used}.")
    except Exception as e:
        flash(f"Training failed: {e}")
    return redirect(url_for("index"))

@app.route("/delete_person", methods=["POST"])
def delete_person():
    global recognizer

    person = request.form.get("person", "").strip()
    if not person:
        flash("No person specified.")
        return redirect(url_for("index"))

    person_dir = os.path.join(CONFIG.FACES_DIR, person)
    if not os.path.isdir(person_dir):
        flash(f"Person '{person}' not found.")
        return redirect(url_for("index"))

    try:
        shutil.rmtree(person_dir)

        remaining = []
        if os.path.isdir(CONFIG.FACES_DIR):
            remaining = [
```



```
        d for d in os.listdir(CONFIG.FACES_DIR)
        if os.path.isdir(os.path.join(CONFIG.FACES_DIR, d))
    ]

    if len(remaining) == 0:
        try:
            if os.path.exists(CONFIG.MODEL_PATH):
                os.remove(CONFIG.MODEL_PATH)
            if os.path.exists(CONFIG.LABELS_PATH):
                os.remove(CONFIG.LABELS_PATH)
        except Exception:
            pass

        with recognizer_lock:
            recognizer = FaceRecognizer(
                CONFIG.MODEL_PATH,
                CONFIG.LABELS_PATH,
                threshold=get_threshold()
            )

        flash(f"Person '{person}' deleted. No persons left, model
cleared.")
        return redirect(url_for("index"))

    total, used = FaceRecognizer.train_from_folder(
        CONFIG.FACES_DIR,
        CONFIG.MODEL_PATH,
        CONFIG.LABELS_PATH
    )

    with recognizer_lock:
        recognizer = FaceRecognizer(
            CONFIG.MODEL_PATH,
            CONFIG.LABELS_PATH,
            threshold=get_threshold()
        )

    flash(f"Person '{person}' deleted. Model retrained (images:
{total}, faces used: {used}).")

    except Exception as e:
        flash(f"Delete failed: {e}")

    return redirect(url_for("index"))

@app.route("/capture", methods=["POST"])
def capture():
    person = request.form.get("person", "").strip()
```



```
n_raw = request.form.get("num", "20")
try:
    n = int(n_raw)
except Exception:
    n = 20

try:
    saved = capture_samples(person, max(5, min(60, n)))
    flash(f"Captured {saved} samples for {person}.")
except Exception as e:
    flash(f"Capture failed: {e}")
return redirect(url_for("index"))

def gen_frames():
    while True:
        try:
            frame = camera.capture_frame()
            ok, buffer = cv2.imencode(".jpg", frame)
            if not ok:
                time.sleep(0.05)
                continue
            yield b"--frame\r\nContent-Type: image/jpeg\r\n\r\n" +
buffer.tobytes() + b"\r\n"
        except Exception:
            time.sleep(0.2)

@app.route("/preview")
def preview():
    return Response(gen_frames(), mimetype="multipart/x-mixed-replace;
boundary=frame")

@app.route("/settings", methods=["GET", "POST"])
def settings():
    if request.method == "POST":
        threshold = request.form.get("threshold",
str(CONFIG.DEFAULT_THRESHOLD)).strip()
        email_enabled = "1" if request.form.get("email_enabled") ==
"on" else "0"
        email_on_granted = "1" if request.form.get("email_on_granted")
== "on" else "0"
        attach_photo = "1" if request.form.get("attach_photo") == "on"
else "0"
        email_cooldown_sec = request.form.get("email_cooldown_sec",
"30").strip()

        db.set_setting("threshold", threshold)
```



```
db.set_setting("email_enabled", email_enabled)
db.set_setting("email_on_granted", email_on_granted)
db.set_setting("attach_photo", attach_photo)
db.set_setting("email_cooldown_sec", email_cooldown_sec)

flash("Settings saved.")
return redirect(url_for("settings"))

current_threshold = db.get_setting("threshold",
str(CONFIG.DEFAULT_THRESHOLD))
email_enabled = (db.get_setting("email_enabled", "1" if
CONFIG.EMAIL_ENABLED_DEFAULT else "0") == "1")
email_on_granted = (db.get_setting("email_on_granted", "0") == "1")
attach_photo = (db.get_setting("attach_photo", "1") == "1")
email_cooldown_sec = db.get_setting("email_cooldown_sec", "30")

return render_template(
    "settings.html",
    threshold=current_threshold,
    email_enabled=email_enabled,
    email_on_granted=email_on_granted,
    attach_photo=attach_photo,
    email_cooldown_sec=email_cooldown_sec,
)

def start_background_threads():
    threading.Thread(target=motion_loop, daemon=True).start()

@app.route("/reset_model", methods=["POST"])
def reset_model():
    global recognizer
    try:
        try:
            if os.path.exists(CONFIG.MODEL_PATH):
                os.remove(CONFIG.MODEL_PATH)
            if os.path.exists(CONFIG.LABELS_PATH):
                os.remove(CONFIG.LABELS_PATH)
        except Exception:
            pass

        with recognizer_lock:
            recognizer = FaceRecognizer(
                CONFIG.MODEL_PATH,
                CONFIG.LABELS_PATH,
                threshold=get_threshold()
            )

    flash("Model reset: lbph.yml and labels.json cleared.")
```



```
except Exception as e:
    flash(f"Reset model failed: {e}")

    return redirect(url_for("settings"))

@app.route("/clear_events", methods=["POST"])
def clear_events():
    try:
        db.clear_events()
        flash("All events cleared.")
    except Exception as e:
        flash(f"Clear events failed: {e}")
    return redirect(url_for("settings"))

if __name__ == "__main__":
    start_background_threads()
    app.run(host="0.0.0.0", port=5000, debug=True, use_reloader=False)
```

Templates/index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Smart Door</title>

<style>
:root{
  --bg:#f7f8fc;
  --card:#ffffff;
  --text:#0f172a;
  --muted:#6b7280;
  --line:#e5e7eb;

  --primary:#2563eb;
  --primary-soft:#eff6ff;

  --ok:#16a34a;
  --ok-bg:#dcfce7;

  --bad:#dc2626;
  --bad-bg:#fee2e2;

  --wait:#64748b;
  --wait-bg:#f1f5f9;

  --radius:22px;
```



```
--shadow:0 20px 45px rgba(15,23,42,.08);
}

*{box-sizing:border-box}

body{
  margin:0;
  font-family: Inter, system-ui, -apple-system, Segoe UI, Roboto, sans-serif;
  background:linear-gradient(180deg,#f7f8fc,#eef2ff);
  color:var(--text);
  padding:36px;
}

h2,h3{margin:0}

.topbar{
  display:flex;
  justify-content:space-between;
  align-items:center;
  margin-bottom:34px;
}

.title h2{
  font-size:28px;
  font-weight:900;
  letter-spacing:.4px;
}

.actions a{
  margin-left:10px;
}

.btn{
  border:0;
  cursor:pointer;
  border-radius:16px;
  padding:10px 18px;
  font-weight:700;
  font-size:14px;
  display:inline-flex;
  align-items:center;
  gap:8px;
  text-decoration:none;
  transition:all .15s ease;
}

.btn.primary{
  background:var(--primary);
```



```
    color:white;
    box-shadow:0 8px 18px rgba(37,99,235,.25);
}
.btn.primary:hover{transform:translateY(-1px)}

.btn.secondary{
    background:#eef2ff;
    color:#1e293b;
}

.layout{
    display:grid;
    grid-template-columns:440px 1fr;
    gap:26px;
}

.card{
    background:var(--card);
    border-radius:var(--radius);
    box-shadow:var(--shadow);
    padding:26px;
}

.card h3{
    font-size:14px;
    font-weight:800;
    color:#334155;
    margin-bottom:16px;
    letter-spacing:.6px;
    text-transform:uppercase;
}

.section{
    margin-top:26px;
    padding-top:26px;
    border-top:1px solid var(--line);
}

.badge{
    display:inline-flex;
    align-items:center;
    padding:8px 18px;
    border-radius:999px;
    font-weight:900;
    font-size:13px;
    letter-spacing:.6px;
}

.ok{background:var(--ok-bg);color:var(--ok)}
```



```
.bad{background:var(--bad-bg);color:var(--bad)}
.wait{background:var(--wait-bg);color:var(--wait)}

.kv{
  display:grid;
  grid-template-columns:120px 1fr;
  gap:14px 18px;
  margin-top:22px;
}

.k{
  font-size:12px;
  color:var(--muted);
  font-weight:700;
  letter-spacing:.4px;
}

.v{
  font-size:15px;
  font-weight:800;
}

input{
  width:100%;
  padding:12px 14px;
  border-radius:14px;
  border:1px solid var(--line);
  font-size:14px;
  margin-top:10px;
}

input:focus{
  outline:none;
  border-color:#93c5fd;
  box-shadow:0 0 0 4px rgba(59,130,246,.15);
}

.people{
  display:grid;
  gap:12px;
}

.person{
  display:flex;
  justify-content:space-between;
  align-items:center;
  padding:12px 16px;
  border-radius:16px;
  background:#f9fafb;
```




```
border:1px solid var(--line);
}

.person b{
  font-size:14px;
}

table{
  width:100%;
  border-collapse:collapse;
}

thead th{
  font-size:12px;
  color:var(--muted);
  text-align:left;
  padding:12px;
  border-bottom:1px solid var(--line);
  letter-spacing:.4px;
}

tbody td{
  padding:14px 12px;
  font-size:13px;
  border-bottom:1px solid #f1f5f9;
}

tbody tr:hover{
  background:#f8fafc;
}

.pill{
  display:inline-flex;
  padding:6px 14px;
  border-radius:999px;
  font-weight:900;
  font-size:12px;
  letter-spacing:.4px;
}

.pill.ok{background:var(--ok-bg);color:var(--ok)}
.pill.bad{background:var(--bad-bg);color:var(--bad)}
.pill.wait{background:var(--wait-bg);color:var(--wait)}
</style>
</head>

<body>

<div class="topbar">
```



```
<div class="title">
  <h2>Smart Door</h2>
</div>
<div class="actions">
  <a class="btn secondary" href="/login">Email</a>
  <a class="btn secondary" href="/settings">Settings</a>
</div>
</div>

<div class="layout">

<!-- LEFT -->
<div class="card">
  <h3>Status</h3>

  <span id="status_badge" class="badge wait">{{ status.last_result
}}</span>

  <div class="kv">
    <div class="k">NAME</div>
    <div class="v" id="last_name">{{ status.last_name or "-" }}</div>

    <div class="k">CONFIDENCE</div>
    <div class="v" id="last_confidence">{{ status.last_confidence or "-"
" }}</div>

    <div class="k">NOTE</div>
    <div class="v" id="last_note">{{ status.note or "-" }}</div>
  </div>

  <div class="section">
    <form action="/trigger" method="post">
      <button class="btn primary" type="submit">Run
Recognition</button>
    </form>
  </div>

  <div class="section">
    <h3>Add / Train Person</h3>

    <form action="/capture" method="post">
      <input name="person" placeholder="Person name" required>
      <input type="number" name="num" value="20" min="5" max="60">
      <button class="btn primary" type="submit" style="margin-
top:14px">
        Capture Samples
      </button>
    </form>
```



```
<form action="/train" method="post" style="margin-top:14px">
  <button class="btn secondary" type="submit">Train Model</button>
</form>
</div>

<div class="section">
  <h3>Authorized Persons</h3>
  <div class="people">
    {% for p in persons %}
    <div class="person">
      <b>{{ p }}</b>
      <form action="/delete_person" method="post">
        <input type="hidden" name="person" value="{{ p }}">
        <button class="btn secondary" type="submit">Delete</button>
      </form>
    </div>
    {% endfor %}
  </div>
</div>
</div>

<!-- RIGHT -->
<div class="card">
  <h3>Recent Events</h3>

  <table>
    <thead>
      <tr>
        <th>TIME</th>
        <th>NAME</th>
        <th>RESULT</th>
        <th>CONF</th>
      </tr>
    </thead>
    <tbody id="events-body">
      {% for e in events %}
      <tr>
        <td>{{ e.ts }}</td>
        <td>{{ e.name }}</td>
        <td>
          <span class="pill {% if e.result=='GRANTED' %}ok{% elif
e.result=='DENIED' %}bad{% else %}wait{% endif %}">
            {{ e.result }}
          </span>
        </td>
        <td>{{ e.confidence }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
```



```
</table>
</div>

</div>

<script>
function poll(){
  fetch("/status_json")
    .then(r=>r.json())
    .then(data=>{
      const s=data.status;
      const badge=document.getElementById("status_badge");

      badge.textContent=s.last_result;
      badge.className="badge "+
        (s.last_result==="GRANTED"? "ok":
        s.last_result==="DENIED"? "bad": "wait");

      document.getElementById("last_name").textContent=s.last_name||"-";
      document.getElementById("last_confidence").textContent=s.last_confidence||"-";
      document.getElementById("last_note").textContent=s.note||"-";

      const tbody=document.getElementById("events-body");
      tbody.innerHTML="";
      data.events.forEach(e=>{
        const row=document.createElement("tr");
        row.innerHTML=`
          <td>${e.ts}</td>
          <td>${e.name||"-"}</td>
          <td><span class="pill
${e.result==="GRANTED"? "ok": e.result==="DENIED"? "bad": "wait"}>${e.result}</span></td>
          <td>${e.confidence||"-"}</td>
        `;
        tbody.appendChild(row);
      });
    });
  setIntervall(poll,2000);
</script>

</body>
</html>
```



Templates/login.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Smart Door - Login</title>

  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      background: #f7f9fc;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    .card {
      background: white;
      padding: 30px;
      width: 420px;
      border-radius: 16px;
      box-shadow: 0 10px 25px rgba(0,0,0,0.05);
      text-align: center;
    }

    h2 {
      margin-bottom: 6px;
    }

    h3 {
      margin-top: 0;
      font-weight: normal;
      color: #555;
    }

    input {
      width: 100%;
      padding: 10px;
      border-radius: 10px;
      border: 1px solid #ccc;
      font-size: 15px;
      margin-top: 8px;
    }

    button {
      margin-top: 14px;
      padding: 10px 14px;
```



```
border-radius: 10px;
border: none;
cursor: pointer;
background: #4f46e5;
color: white;
font-size: 14px;
width: 100%;
}

.secondary {
background: #e5e7eb;
color: #111;
}

.danger {
background: #dc2626;
}

.muted {
color: #777;
font-size: 13px;
margin-top: 10px;
}

.flash {
background: #eef2ff;
border: 1px solid #c7d2fe;
color: #3730a3;
padding: 10px;
border-radius: 10px;
margin-bottom: 14px;
font-size: 14px;
}

.current {
margin-top: 12px;
font-size: 14px;
color: #333;
background: #f3f4f6;
padding: 10px;
border-radius: 10px;
}

.divider {
margin: 18px 0;
border: none;
border-top: 1px solid #eee;
}
</style>
```



```
</head>

<body>

<div class="card">

  <h2>Smart Door</h2>
  <h3>Email Notifications</h3>

  {% with messages = get_flashed_messages() %}
  {% if messages %}
    <div class="flash">
      {% for m in messages %}
        <div>{{ m }}</div>
      {% endfor %}
    </div>
  {% endif %}
  {% endwith %}

  <form method="post">
    <input type="email"
      name="email"
      placeholder="ellenmavrogianni@gmail.com"
      value="{{ current_email }}"
      required>
    <button type="submit">Save email</button>
  </form>

  <div class="muted">
    This email will receive Access Denied alerts and optionally Access
    Granted events.
  </div>

  {% if current_email %}
    <div class="current">
      Current email:<br>
      <strong>{{ current_email }}</strong>
    </div>

    <hr class="divider">

    <form method="post" action="/logout">
      <button class="danger">Logout and remove email</button>
    </form>
  {% endif %}

</div>

</body>
```



</html>

Templates/settings.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Smart Door - Settings</title>

  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 24px;
      background: #f7f9fc;
    }

    h2 {
      margin-bottom: 12px;
    }

    .card {
      max-width: 720px;
      background: white;
      border-radius: 14px;
      padding: 24px;
      box-shadow: 0 10px 25px rgba(0,0,0,0.05);
    }

    .row {
      margin-bottom: 18px;
    }

    label {
      font-weight: bold;
      display: block;
      margin-bottom: 6px;
    }

    input[type="number"] {
      padding: 8px;
      width: 120px;
      border-radius: 8px;
      border: 1px solid #ccc;
      font-size: 15px;
    }

    .hint {
      font-size: 13px;
```




```
    color: #666;
    margin-top: 4px;
}

.checkbox-row {
    display: flex;
    align-items: center;
    gap: 10px;
    margin-bottom: 12px;
}

.checkbox-row input {
    transform: scale(1.2);
}

.actions {
    margin-top: 20px;
    display: flex;
    gap: 12px;
}

button {
    padding: 10px 16px;
    border-radius: 10px;
    border: none;
    cursor: pointer;
    font-size: 14px;
    background: #4f46e5;
    color: white;
}

button.secondary {
    background: #e5e7eb;
    color: #111;
}

.danger {
    background: #dc2626;
}

.danger-secondary {
    background: #b91c1c;
}

.divider {
    margin: 26px 0;
    border: none;
    border-top: 1px solid #eee;
}
```



```
.section-title {
  font-size: 18px;
  font-weight: bold;
  margin-bottom: 10px;
}

.danger-box {
  border: 1px solid #fecaca;
  background: #fff7f7;
  padding: 18px;
  border-radius: 14px;
}
</style>
</head>

<body>

<h2>Settings</h2>

<div class="card">

  <form method="post">

    <div class="row">
      <label>Recognition threshold</label>
      <input type="number" name="threshold" value="{{ threshold }}"
step="1">
      <div class="hint">
        Lower usually accepts more matches. Tune based on your tests.
      </div>
    </div>

    <hr class="divider">

    <div class="row checkbox-row">
      <input type="checkbox" name="email_enabled" {% if email_enabled
}%checked{% endif %}>
      <label style="margin:0;">Enable email notifications</label>
    </div>

    <div class="row checkbox-row">
      <input type="checkbox" name="email_on_granted" {% if
email_on_granted %}>checked{% endif %}>
      <label style="margin:0;">Send email also on Access
Granted</label>
    </div>

    <div class="row checkbox-row">
```



```
<input type="checkbox" name="attach_photo" {% if attach_photo
}%checked{% endif %}>
<label style="margin:0;">Attach photo to emails</label>
</div>

<div class="row">
  <label>Email cooldown (seconds)</label>
  <input type="number" name="email_cooldown_sec" value="{{
email_cooldown_sec }}">
  <div class="hint">
    0 means no cooldown. Example: 30 sends max 1 email per 30
seconds.
  </div>
</div>

<div class="actions">
  <button type="submit">Save</button>
  <a href="/" style="text-decoration:none;">
    <button type="button" class="secondary">Back</button>
  </a>
</div>

<hr class="divider">

<div class="danger-box">
  <div class="section-title" style="color:#991b1b;">Danger
zone</div>
  <div class="hint" style="margin-bottom: 14px;">
    These actions permanently delete data.
  </div>

  <div class="actions">
    <button class="danger"
      type="submit"
      formaction="/reset_model"
      formmethod="post"
      onclick="return confirm('Reset model? This will delete
all trained faces.');">
      Reset model
    </button>

    <button class="danger-secondary"
      type="submit"
      formaction="/clear_events"
      formmethod="post"
      onclick="return confirm('Clear all events? This will
delete the event history.');">
      Clear events
    </button>
```



```
</div>  
</div>  
  
</form>  
  
</div>  
  
</body>  
</html>
```