

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



Πτυχιακή Εργασία

***“Ανάπτυξη εφαρμογής διαχείρισης
μητρώου ασθενών με διαβήτη”***

Σπυρίδωνος Ξιάρχου Π2015048

Επιβλέπων Καθηγητής: Θεμιστοκλής Έξαρχος

30 Σεπτεμβρίου 2023

Επιβλέπων Καθηγητής

Θεμιστοκλής Έξαρχος, Επίκουρος Καθηγητής, Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Θεμιστοκλής Έξαρχος, Επίκουρος Καθηγητής, Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Αννα Σωτηροπούλου, ΕΔΙΠ Τμήματος Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Κάτια - Αήδα Κερμανίδου, Αναπληρώτρια Καθηγήτρια, Τμήμα Πληροφορικής,
Ιόνιο Πανεπιστήμιο

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως απώτερο σκοπό την πρόταση εφαρμογής διαχείρισης μητρώων ασθενών που πάσχουν από διαβήτη.

Στην αρχή γίνεται παρουσίαση του προβλήματος και των συνιστωσών του, σε συνδυασμό με παρουσίαση των υπαρχόντων μεθόδων αποθήκευσης, οι οποίες μας οδήγησαν στην πρόταση ενός νέου συστήματος.

Μετέπειτα γίνεται λεπτομερής καταγραφή των λειτουργιών της εφαρμογής, δίνοντας ιδιαίτερη έμφαση στα αποσπάσματα που αναφέρονται στην καταγραφή των μητρώων και την αποθήκευση των ιατρικών αρχείων. Επιπλέον, παρατίθενται όλα τα σημαντικά εργαλεία που αξιοποιήθηκαν κατά την περίοδο της ανάπτυξης της εφαρμογής.

Στη συνέχεια ακολουθεί πρόταση μελλοντικών προεκτάσεων οι οποίες περιλαμβάνουν σημαντικές βελτιώσεις σε περίπτωση μελλοντικής χρήσης της εφαρμογής, αλλά και ένα υποκεφάλαιο που περιγράφει συνοπτικά τα βήματα ανάπτυξης της εφαρμογής. Εν κατακλείδι, γίνεται παρουσίαση των συμπερασμάτων που προέκυψαν καθ' όλη τη διάρκεια εκπόνησης της πτυχιακής εργασίας.

Πρόλογος και ευχαριστίες

Θεωρώ στοιχειώδη υποχρέωσή μου, να εκφράσω αρχικά τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου στο Τμήμα Πληροφορικής του Ιονίου Πανεπιστημίου, κ. Θεμιστοκλή Έξαρχο, για το χρόνο που διέθεσε και την καθοδήγησή του με οδηγίες και χρήσιμες συμβουλές, για την ολοκλήρωση της πτυχιακής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Τμήματος Πληροφορικής για τη συμβολή τους -κατά τα χρόνια φοίτησής μου στο Ιόνιο Πανεπιστήμιο- στην επιστημονική και τεχνολογική μου συγκρότηση.

Ένα ακόμα “ευχαριστώ” οφείλω σε όλους όσοι συνέβαλαν με κάθε δόκιμο τρόπο στην ολοκλήρωση της πτυχιακής μου εργασίας μου.

Τέλος, εκφράζω τις ευχαριστίες μου στους γονείς μου για την υποστήριξη τους (οικονομική και ηθική), καθ’ όλο το διάστημα των σπουδών μου.

Περιεχόμενα

<u>Περίληψη</u>	<u>4</u>
<u>Πρόλογος και ευχαριστίες</u>	<u>5</u>
<u>Περιεχόμενα</u>	<u>6</u>
<u>ΕΙΣΑΓΩΓΗ</u>	<u>7</u>
<u>A. ΔΙΑΒΗΤΗΣ: ΟΙ ΑΙΤΙΕΣ ΚΑΙ Η ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ</u>	<u>8</u>
<u>A.1 Τύποι διαβήτη</u>	<u>8</u>
<u>A.2 Αίτια διαβήτη</u>	<u>8</u>
<u>A.3 Διάγνωση & Αντιμετώπιση</u>	<u>11</u>
<u>A.4 Δεδομένα</u>	<u>13</u>
<u>B. Η ΕΦΑΡΜΟΓΗ ΚΑΙ Η ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΤΗΣ</u>	<u>16</u>
<u>B.1 Προτεινόμενο σύστημα</u>	<u>16</u>
<u>B.2 Πλεονεκτήματα εφαρμογής</u>	<u>20</u>
<u>B.3 Προϋπάρχουσες Υλοποιήσεις και Μελέτες</u>	<u>21</u>
<u>Γ. ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΑΞΙΟΠΟΙΗΘΗΚΑΝ</u>	<u>25</u>
<u>Δ. ΟΘΟΝΕΣ</u>	<u>30</u>
<u>Δ.1. Οθόνες Εφαρμογής</u>	<u>30</u>
<u>Δ.2 Οθόνες Firebase</u>	<u>35</u>
<u>Ε. ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ</u>	<u>38</u>
<u>E.1 Προσθήκη 2FA για πρόσβαση στην εφαρμογή</u>	<u>38</u>
<u>E.2 Προσθήκη Αποκεντροποιημένης Βάσης Δεδομένων</u>	<u>38</u>
<u>E.3 Σύνδεση Εφαρμογής με Wearables</u>	<u>38</u>
<u>E.4 Δημιουργία Πλατφόρμας Διαμοιρασμού Δεδομένων</u>	<u>39</u>
<u>E.5 Δημιουργία Εφαρμογής για Ios, Android</u>	<u>39</u>
<u>E.6 Υλοποίηση Φόρμας Παροχής Συναίνεσης για Επεξεργασία Δεδομένων</u>	<u>39</u>
<u>ΣΥΜΠΕΡΑΣΜΑΤΑ</u>	<u>40</u>
<u>Programming Journey</u>	<u>41</u>
<u>REFERENCES</u>	<u>43</u>
<u>APPENDIX</u>	<u>44</u>

ΕΙΣΑΓΩΓΗ

Ο διαβήτης αποτελεί μεταβολική διαταραχή, η οποία συμβαίνει όταν το ανθρώπινο σώμα αδυνατεί να παράγει ή να χρησιμοποιήσει την ουσία Ινσουλίνη με αποτελεσματικότητα^[1]. Η Ινσουλίνη αποτελεί ορμόνη η οποία παράγεται από το πάγκρεας και ρυθμίζει τα επίπεδα Σακχάρου στο αίμα^[2]. Ο διαβήτης έχει εξελιχθεί σε μια παγκόσμια πανδημία καθώς, σύμφωνα με τη Παγκόσμια Ομοσπονδία Διαβήτη, κατά το έτος 2019, υπήρχαν 463 εκατομμύρια άνθρωποι που ζούσαν με τη πάθηση, ενώ ταυτόχρονα ο αριθμός υπολογίζεται ότι θα ξεπεράσει τα 700 εκατομμύρια μέχρι το έτος 2045^[3]. Ταυτόχρονα, έχει αναδειχθεί και αποδειχθεί μέσα από μελέτες ότι η αύξηση των περιστατικών διαβήτη συνδέεται με αυξήσεις στα ποσοστά παχυσαρκίας, την καθιστική ζωή και τις ανθυγιεινές διατροφικές συνήθειες του παγκόσμιου πληθυσμού^[4].

Μέσω της εργασίας μας θα προβούμε στη συνέχεια, σε παρουσίαση της νόσου του διαβήτη, των συνεπειών της, τους λόγους για τους οποίους κάποιος μπορεί να εμφανίσει προδιάθεση ή και την ίδια την πάθηση, αλλά και των μεθόδων διάγνωσης και αντιμετώπισης της ασθένειας.

Καθώς, αφενός, ο διαβήτης απαιτεί προσεκτική διαχείριση και παρακολούθηση των βιοδεικτών των πασχόντων και αφετέρου, οι παραδοσιακές μέθοδοι διατήρησης φυσικών αντιτύπων τείνουν να αποδεικνύονται προβληματικές, εξ αιτίας λανθασμένων καταχωρήσεων ή καταστροφής τους, για την εξάλειψη αυτών των προβλημάτων -αξιοποιώντας μια συλλογή τεχνολογικών εργαλείων-, ανέπτυξα την εφαρμογή «Diabetic Data Solutions», που προορίζεται για χρήση από επαγγελματίες υγείας.

Πρόκειται για ένα σύστημα διαχείρισης μητρώων ασθενών, μια εφαρμογή αποθήκευσης και διαχείρισης δεδομένων, που προάγει τη συνεργασία μεταξύ ασθενών, ιατρικού προσωπικού και οργανισμών, καθώς διαθέτει τη δυνατότητα εξαγωγής και διαμοιρασμού των δεδομένων.

Ταυτόχρονα, όλα τα εξαγόμενα δεδομένα δύναται να χρησιμοποιηθούν για ερευνητικούς σκοπούς από ιατρικούς οργανισμούς αλλά και αυτόνομους επαγγελματίες στο χώρο της υγείας, συνεισφέροντας στην περαιτέρω μελέτη και αντιμετώπιση της νόσου.

A. ΔΙΑΒΗΤΗΣ: ΟΙ ΑΙΤΙΕΣ ΚΑΙ Η ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ

A.1 Τύποι διαβήτη

Η πάθηση του διαβήτη συνήθως διαιρείται σε δύο επί μέρους παραλλαγές: τον διαβήτη τύπου 1 και 2. Στη περίπτωση του διαβήτη τύπου 1, το ανοσοποιητικό σύστημα του ασθενούς επιτίθεται στα κύτταρα που ευθύνονται για την παραγωγή ινσουλίνης, οδηγώντας σε πλήρη απουσία παραγωγής, γεγονός που ενδέχεται να επιφέρει σημαντικές συνέπειες στην υγεία του ασθενούς. Στον διαβήτη τύπου 2, ο οποίος αποτελεί έως και το 90% των περιπτώσεων, ο οργανισμός του ασθενούς αποκτά ανθεκτικότητα ενάντια στη δράση της ινσουλίνης ή όταν το πάγκρεας του ασθενούς αποτυγχάνει να παράγει αρκετή ποσότητα ινσουλίνης έτσι ώστε να καλύψει τις ανάγκες του οργανισμού.^[1]

A.2 Αίτια διαβήτη

Τα αίτια του διαβήτη αποτελούν ένα πολυπαραγοντικό και σύνθετο ζήτημα, καθώς επηρεάζονται από περιβαλλοντικούς και γενετικούς παράγοντες. Παρακάτω παρατίθενται τα κυριότερα αίτια^[4]:

- **Γενετικοί παράγοντες**

Σε περιπτώσεις όπου μέλη της οικογένειας ενός ασθενούς πάσχουν από οποιοδήποτε τύπο διαβήτη, έχουν παρατηρηθεί αυξημένα ποσοστά προδιάθεσης για τον ίδιο τον ασθενή. Συνεπώς, η διατήρηση οικογενειακού ιστορικού αναφορικά με το διαβήτη ενδέχεται να συνεισφέρει στην έγκαιρη πρόληψη, διαχείριση και αντιμετώπιση της ασθένειας.



Εικόνα 1 Η διπλή έλικα του DNA

- ***Παχυσαρκία***

Η παχυσαρκία είναι από τους πιο συνήθεις παράγοντες κινδύνου που οδηγούν στην ανάπτυξη διαβήτη τύπου 2. Η υπέρμετρη αύξηση σωματικού λίπους ενδέχεται να προκαλέσει αδυναμία αποτελεσματικής χρήσης της ινσουλίνης από τον οργανισμό, αύξηση ανθεκτικότητας ενάντια στην ινσουλίνη και αύξηση του επιπέδου Σακχάρου στο αίμα.

- ***Απουσία σωματικής άσκησης***

Η απουσία σωματική άσκησης επίσης συμβάλλει στον κίνδυνο ανάπτυξης διαβήτη τύπου 2. Η σωματική αύξηση αυξάνει την ευαισθησία του οργανισμού στην ινσουλίνη, γεγονός που οδηγεί σε καλύτερη ρύθμιση των επιπέδων σακχάρου.

- ***Ανθυγιεινή διατροφή***

Η κατανάλωση τροφών με υψηλή περιεκτικότητα σε ζάχαρη, κορεσμένα λίπη η κατανάλωση επεξεργασμένων τροφών αυξάνει το κίνδυνο ανάπτυξης διαβήτη τύπου 2. Η κατανάλωση των προαναφερθέντων τροφών ενδέχεται να οδηγήσει σε παχυσαρκία και ανθεκτικότητα προς την ινσουλίνη, γεγονότα τα οποία συνεισφέρουν στην εκδήλωση της ασθένειας.

- ***Ηλικία***

Ο κίνδυνος ανάπτυξης διαβήτη τύπου 2 αυξάνεται κατά τη γήρανση καθώς ο ανθρώπινος οργανισμός καθίσταται λιγότερο ευαίσθητος στην ινσουλίνη, με αποτέλεσμα να παρουσιάζει δυσκολίες στον έλεγχο των επιπέδων σακχάρου στο αίμα.

- ***Εγκυμοσύνη***

Ο διαβήτης κυοφορίας είναι ένας τύπος διαβήτη ο οποίος λαμβάνει χώρα κατά τη διάρκεια εγκυμοσύνης και οφείλεται σε ορμονικές διαταραχές οι οποίες δυσχεραίνουν τον έλεγχο επιπέδων σακχάρου στο αίμα.

- ***Ιατρικοί παράγοντες***

Έχει παρατηρηθεί ότι συγκεκριμένες ασθένειες, όπως για παράδειγμα το Σύνδρομο Πολυκυστικών Ωοθηκών, αυξάνουν κατακόρυφα τον κίνδυνο εκδήλωσης διαβήτη τύπου 2.

- ***Φαρμακευτικές αγωγές***

Η χρήση φαρμακευτικών αγωγών όπως κορτικοστεροειδών, τα οποία οδηγούν σε αύξηση των επιπέδων σακχάρου, συνεισφέρει στην αύξηση του κινδύνου ανάπτυξης διαβήτη.

Αξίζει να σημειωθεί ότι ενώ οι παραπάνω παράγοντες κινδύνου αυξάνουν τον κίνδυνο εμφάνισης διαβήτη, δεν είναι απαραίτητο ότι κάποιος ασθενής με ένα ή περισσότερα από τα προαναφερθέντα χαρακτηριστικά θα αναπτύξει την ασθένεια. Η ύπαρξη επιπλέον παραγόντων που συνεισφέρουν στην ανάπτυξη της ασθένειας, δεν αποκλείεται, καθώς χρειάζεται περισσότερη έρευνα για την ανάδειξη των πλήρων αιτιών.

Α.3 Διάγνωση και αντιμετώπιση

Κίνδυνοι διαβήτη

Οι συνέπειες αποτυχίας ελέγχου του διαβήτη ενδέχεται να είναι σοβαρές, ακόμα και απειλητικές για τη ζωή του ασθενούς. Στις συνέπειες περιλαμβάνονται καρδιαγγειακές ασθένειες, νεφροπάθεια, τύφλωση, ακρωτηριασμοί και άλλα. Ο διαβήτης έχει συσχετιστεί με μείωση του προσδόκιμου ορίου ζωής και χαμηλότερο βιοτικό επίπεδο.

Διάγνωση διαβήτη

Η διάγνωση του διαβήτη γίνεται με τους ακόλουθους τρόπους:

- **Βιοδείκτες και εξετάσεις**

Για τη διάγνωση και πρόληψη του διαβήτη αξιοποιούνται βιοδείκτες του εκάστοτε ασθενούς, των οποίων οι τιμές ενδέχεται να υποδεικνύουν την ανάπτυξη διαβήτη. Παρακάτω παρατίθεται πίνακας των εξετάσεων, καθώς και των τιμών που υποδηλώνουν διαβήτη^[6].

Εξετάσεις	Ανώτατα όρια
Έλεγχος γλυκόζης στο πλάσμα μετά από νηστεία	>115 mg/dl
Έλεγχος ανοχής γλυκόζης από το στόμα	>170 mg/dl (1 ώρα από την λήψη) >120 mg/dl (2 ώρες από την λήψη)
Έλεγχος Αιμοσφαιρίνης A1C	>6%

Αξίζει να σημειωθεί ότι, σε συνδυασμό με τους προαναφερθέντες βιοδείκτες, το θεράπον ιατρικό προσωπικό δύναται να συμβουλευτεί και άλλους, όπως η μέτρηση των κετονών στα ούρα ή και τα επίπεδα ινσουλίνης στο αίμα. Ταυτόχρονα ενδέχεται να ζητηθούν επιπλέον γενετικές εξετάσεις, έτσι ώστε να εξακριβωθεί η ύπαρξη προδιάθεσης στην ασθένεια.

- **Έλεγχος αντιγόνων λευκοκυττάρων**

Η εξέταση αντιγόνων λευκοκυττάρων εξετάζει συγκεκριμένα γονίδια που έχουν συσχετιστεί με την ανάπτυξη διαβήτη τύπου 1 και συνίσταται για ασθενείς οι οποίοι έχουν συγγενείς που πάσχουν από την ασθένεια ή που παρουσιάζουν παράγοντες κινδύνου της.

- ***Έλεγχος για μονογονιδιακό Σακχαρώδη διαβήτη***

Ο μονογονιδιακός Σακχαρώδης διαβήτης αποτελεί σπάνια μορφή διαβήτη και προκαλείται από τη μετάλλαξη ενός και μόνο γονιδίου. Με την αναγνώριση του γονιδίου υποβοηθάται η διαδικασία θεραπείας και διαχείρισης της ασθένειας.

- ***Εκτίμηση Πολυγονιαδικού κινδύνου***

Η Εκτίμηση Πολυγονιαδικού κινδύνου αποτελεί σχετικά νέα εξέταση κατά την οποία αξιοποιούνται πληροφορίες από παραλλαγές γονιδίων, έτσι ώστε να υπολογιστεί το ποσοστό κινδύνου του ασθενούς για ανάπτυξη διαβήτη τύπου 2. Η εξέταση συνίσταται για ασθενείς οι οποίοι έχουν συγγενείς που πάσχουν από την ασθένεια ή που παρουσιάζουν παράγοντες κινδύνου της.

Αντιμετώπιση του διαβήτη

Η διαχείριση του διαβήτη αποτελεί μια ισόβια προσπάθεια για τους ασθενείς και απαιτεί εμπεριστατωμένη προσέγγιση. Ο στόχος της διαχείρισης διαβήτη είναι να διατηρεί τα επίπεδα σακχάρου στο αίμα ελεγχόμενα, για την αποτροπή επιπλοκών και συνεπώς την προστασία της ζωής του ασθενούς^[5].

1) Υγιεινή διατροφή

Η υγιεινή διατροφή παρουσιάζεται ως αναπόσπαστο κομμάτι της διαχείρισης διαβήτη. Μια διατροφή που περιλαμβάνει τροφές ολικής άλεσης, φρούτα, λαχανικά, άπαχες πρωτεΐνες και υγιεινά λίπη ενδέχεται να συμβάλλει στον έλεγχο του επιπέδου σακχάρου ενώ συνεισφέρει στη γενική υγεία του οργανισμού. Παράλληλα, κρίνεται απαραίτητη η μείωση κατανάλωσης επεξεργασμένων τροφών και τροφών με κορεσμένα λίπη ή υψηλή περιεκτικότητα σε ζάχαρη.

2) Συχνή σωματική άσκηση

Όπως προαναφέρθηκε η σωματική άσκηση συνεισφέρει στον περιορισμό του διαβήτη καθώς συνεισφέρει στην μείωση ανοχής του οργανισμού στην ινσουλίνη και μείωση των επιπέδων σακχάρου στο αίμα. Συνιστανται 150 λεπτά άσκησης μέτριας έντασης, ημερησίως, όπως, ποδηλασία, κολύμβηση ή ζωηρό περπάτημα.

3) Φαρμακευτική αγωγή

Ανάλογα με τον τύπο του διαβήτη, ενδέχεται να κριθεί απαραίτητη η χορήγηση φαρμακευτικής αγωγής. Στην περίπτωση του διαβήτη τύπου 1, συνήθως απαιτείται είτε αντλία παροχής ινσουλίνης ή ενέσεις ινσουλίνης. Στη περίπτωση του διαβήτη τύπου 2, χορηγούνται φαρμακευτικές αγωγές όπως μετφορμίνη, σουλφονυλουρίες και GLP-1 receptor agonists.

4) Έλεγχος επιπέδων σακχάρου

Ο συνεχής έλεγχος των επιπέδων σακχάρου αποτελεί κρίσιμο εργαλείο στην διαχείριση της ασθένειας. Οι ασθενείς οφείλουν να διεξάγουν τους ελέγχους αυτούς πολλαπλές φορές ανά ημέρα, κυρίως πριν από γεύματα και πριν τον ύπνο. Συχνές ιατρικές εξετάσεις

5) Αιματολογικές εξετάσεις

Για τον πρόωρο εντοπισμό του φαινομένου, συνίστανται συχνές αιματολογικές εξετάσεις, έτσι ώστε η ασθένεια να αντιμετωπιστεί εγκαίρως.

6) Διακοπή καπνίσματος

Το κάπνισμα δύναται να επιδεινώσει τις συνέπειες του διαβήτη σε περιπτώσεις καρδιαγγειακών ασθενειών ή νεφροπαθειών.

7) Διαχείριση άγχους

Το άγχος επηρεάζει τα επίπεδα σακχάρου στο αίμα, επομένως συνίσταται η διαχείριση του μέσω τεχνικών χαλάρωσης όπως ο διαλογισμός.

A.4 Δεδομένα

Κομβικότητα των δεδομένων

Σύμφωνα με τα προηγούμενα, τα δεδομένα των ασθενών αποτελούν τον θεμέλιο λίθο όσον αφορά την καταπολέμηση και διαχείριση του διαβήτη. Το ιατρικό προσωπικό βασίζεται σε αυτά, έτσι ώστε να λάβει εμπεριστατωμένες αποφάσεις σχετικά με τη διάγνωση, διαχείριση και αντιμετώπιση της νόσου. Σε περίπτωση απουσίας επαρκών δεδομένων, η δυνατότητα παροχής ποιοτικών και στοχευμένων υπηρεσιών περίθαλψης θα δυσχεραίνονταν σημαντικά.

Συλλεγόμενα δεδομένα

Το θεράπων ιατρικό προσωπικό συλλέγει πληθώρα δεδομένων που σχετίζονται με τους ασθενείς, τα οποία αξιοποιούνται για τη λήψη αποφάσεων σε σχέση με την ασθένεια. Στα συλλεγόμενα δεδομένα περιλαμβάνονται δημογραφικά στοιχεία, τα οποία ενδέχεται να υποδείξουν τον παράγοντα κινδύνου του εκάστοτε ασθενούς, το ιατρικό του ασθενούς, από το οποίο δύναται να αναγνωριστούν πιθανές μελλοντικές επιπλοκές της ασθένειας, εργαστηριακά αποτελέσματα, όπως μετρήσεις σακχάρου και γλυκοζυλιωμένης, οι οποίες βοηθούν στον έλεγχο προόδου της ασθένειας και στην προσαρμογή της θεραπείας ανάλογα με την περίπτωση.

Τέλος, δεδομένα που αφορούν τον τρόπο ζωής και τα λαμβανόμενα φάρμακα του ασθενούς, ενδέχεται να υποδείξουν πιθανές επιπλοκές αλλά και να παρέχουν υπηρεσίες εξατομικευμένης υγείας^[7].

Ηλεκτρονικά Αρχεία Υγείας

Τα Ηλεκτρονικά Αρχεία Υγείας (Electronic Health Records) αποτυπώνουν όλα τα σχετικά με τον ασθενή συλλεγόμενα δεδομένα, ενώ ουσιαστικά αποτελούν πρότυπο αποθήκευσης ιατρικών πληροφοριών. Τα EHR προάγουν την απρόσκοπτη καταγραφή, διατήρηση και διαφύλαξη των δεδομένων, ενώ επίσης διευκολύνουν την μεταφορά τους μεταξύ ιατρικών οργανισμών. Ταυτόχρονα όμως, έχουν παρατηρηθεί προκλήσεις κατά την ενσωμάτωση και χρήση συστημάτων EHR. Οι προκλήσεις αυτές αφορούν την ιδιωτικότητα των ασθενών και την ασφάλεια των δεδομένων τους, την διαλειτουργικότητα των συστημάτων μεταξύ υπηρεσιών και την απαραίτητη εκπαίδευση που απαιτείται έτσι ώστε το ιατρικό προσωπικό να καταφέρει να ενσωματώσει τα συστήματα στη ροή εργασιών τους, η οποία επίσης συνοδεύεται από το απαραίτητο κόστος^[8].

Ιδιωτικότητα και ασφάλεια

Η ιδιωτικότητα και ασφάλεια των ιατρικών δεδομένων αποτελεί πρωτεύων ζήτημα στην Ευρώπη, ειδικά μετά την εμφάνιση του GDPR το 2018. Ο GDPR αποτελεί ευρωπαϊκό κανονισμό και δεσμεύει όλους τους οργανισμούς που επεξεργάζονται προσωπικά δεδομένων πολιτών^[9].

Στην περίπτωση του διαβήτη, οι ιατρικοί οργανισμοί οφείλουν να είναι βέβαιοι ότι κάθε συλλογή, επεξεργασία και αποθήκευση δεδομένων υπακούει στο GDPR, ενώ επίσης ότι τηρούνται κανονισμοί, όπως η συλλογή δεδομένων μόνο μετά από ρητή συναίνεση του πολίτη, αλλά και η διατήρηση των δικαιωμάτων του πολίτη για προβολή, επεξεργασία και διαγραφή των προσωπικών δεδομένων του.

Επιπλέον, ο GDPR ορίζει ότι οι πάροχοι ιατρικών υπηρεσιών οφείλουν να αναφέρουν οποιεσδήποτε παραβιάσεις αρχείων στις αρμόδιες αρχές, έως και 72 ώρες μετά από την στιγμή που αντιλήφθηκαν την παραβίαση. Αποτυχία συμμόρφωσης ενδέχεται να επιφέρει σημαντικά χρηματικά πρόστιμα^[9].

Ανάλυση δεδομένων και υποστήριξη αποφάσεων

Τα προαναφερθέντα δεδομένα, μετά από συναίνεση των ασθενών και ανωνυμοποίηση τους, μπορούν να αξιοποιηθούν σε ιατρικές έρευνες, οι οποίες διεξάγονται από δημόσιους ή ιδιωτικούς οργανισμούς. Συνεπώς, παρέχονται κίνητρα στους γιατρούς να διατηρούν σωστές και ακριβείς καταγραφές μητρώων, αλλά και στους ασθενείς, στους οποίους δίνεται η δυνατότητα να συνεισφέρουν στο έργο της επιστήμης, παρέχοντας μια απλή συγκατάθεση.

B. Η ΕΦΑΡΜΟΓΗ ΚΑΙ Η ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΤΗΣ

B.1 Προτεινόμενο Σύστημα

Η διαχείριση του διαβήτη αποτελεί μια δύσκολη πρόκληση για τους ασθενείς, καθώς απαιτεί προσεκτική διαχείριση και παρακολούθηση των βιοδεικτών τους. Οι παραδοσιακές μέθοδοι διατήρησης φυσικών αντιτύπων τείνουν να αποδεικνύονται προβληματικές, καθώς ενδέχεται να αποτύχουν εξ αιτίας λανθασμένων καταχωρήσεων ή καταστροφής τους. Για την εξάλειψη αυτών των προβλημάτων, αναπτύξαμε την εφαρμογή «Diabetic Data Solutions» η οποία προορίζεται για χρήση από επαγγελματίες υγείας.

Αξιοποιώντας μια συλλογή τεχνολογικών εργαλείων, αναπτύξαμε μια εφαρμογή η οποία είναι ικανή για αποθήκευση και διαχείριση δεδομένων, σύστημα διαχείρισης μητρώων ασθενών, ενώ ταυτόχρονα παρέχει δυνατότητες διαμοιρασμού των δεδομένων. Παρακάτω θα προχωρήσουμε σε ανάλυση όλων των λειτουργικοτήτων της εφαρμογής.

Εγγραφή και Είσοδος χρήστη

Κατά την είσοδο στην εφαρμογή, δίνεται στους χρήστες οι επιλογές εγγραφής ή εισόδου στην εφαρμογή. Σε περίπτωση που ο χρήστης δεν έχει λογαριασμό, μπορεί να προχωρήσει σε δημιουργία λογαριασμού, παρέχοντας το email του και τον κωδικό που επιθυμεί. Σε περίπτωση επιτυχημένης εγγραφής, οδηγείται κατευθείαν στην οθόνη επιλογής λειτουργικότητας, αλλιώς εμφανίζεται κατάλληλο μήνυμα αποτυχημένης εγγραφής. Σε περίπτωση που ο χρήστης έχει ήδη λογαριασμό, μπορεί να χρησιμοποιήσει τα στοιχεία του για να εισέλθει στην εφαρμογή.

Σελίδα υποδοχής

Σε περίπτωση επιτυχημένης εισόδου, ο χρήστης οδηγείται στη σελίδα υποδοχής, όπου δύναται να επιλέξει ανάμεσα σε δύο επιλογές: Drive και Registry. Η κάθε επιλογή περιλαμβάνει διαφορετικές λειτουργικότητες της εφαρμογής. Παρακάτω θα προβούμε σε περαιτέρω ανάλυση των δύο επιλογών, καθώς ουσιαστικά αντιπροσωπεύουν ένα μεγάλο τμήμα της εφαρμογής.

Αυθεντικοποίηση

Για την αυθεντικοποίηση των χρηστών αξιοποιείται η λειτουργικότητα που προσφέρεται από το Firebase. Συνδέουμε το διαδικτυακό περιβάλλον του Firebase με την εφαρμογή μας και μετά από σχετική παραμετροποίηση έχουμε έναν ασφαλή και αξιόπιστο τρόπο αυθεντικοποίησης.

Χώρος αποθήκευσης (Drive)

Επιλέγοντας το εικονίδιο με όνομα Drive, ο χρήστης μεταφέρεται σε διαφορετική σελίδα, όπου του παρέχονται οι δυνατότητες δημιουργίας φακέλου, μεταφόρτωσης αρχείου, επιστροφή στη σελίδα υποδοχής και τέλος αποσύνδεση από την εφαρμογή.

Μεταφόρτωση

Σε περίπτωση επιλογής μεταφόρτωσης αρχείου, ο χρήστης καλείται να επιλέξει το αρχείο προς μεταφόρτωση από τη συσκευή του. Μετά την επιλογή, εμφανίζεται μια μπάρα προόδου, η οποία όταν γεμίσει σηματοδοτεί και την επιτυχή μεταφόρτωση και εμφάνιση του αρχείου στην οθόνη.

Δημιουργία και χρήση φακέλων

Σε περίπτωση επιλογής δημιουργίας φακέλου, μετά την ονομασία του, δημιουργείται ένας φάκελος στην οθόνη μέσα στον οποίο ο χρήστης μπορεί να πλοηγηθεί, εφόσον τον επιλέξει. Μέσα στο περιβάλλον του φακέλου, ο χρήστης μπορεί να προβεί σε μεταφόρτωση αρχείων, να επιλέξει να μεταβεί στην προηγούμενη σελίδα, ή να προβάλλει ήδη αποθηκευμένα αρχεία. Σε περίπτωση επιλογής προβολής εικόνων, αυτές προβάλλονται σε αναδυόμενο παράθυρο, ενώ σε περίπτωση αρχείων όπως csv ή pdf, γίνεται λήψη τους στη συσκευή του χρήστη.

Επεξεργασία καταχωρήσεων και τύποι αρχείων

Στις περιπτώσεις αρχείων και φακέλων, με την χρήση του δεξιού “κλικ” πάνω στο εικονίδιό τους, δίνεται στον χρήστη η δυνατότητα μετονομασίας και διαγραφής τους. Συμπληρωματικά, η εφαρμογή επιτρέπει την μεταφόρτωση πολλαπλών τύπων αρχείων, ενώ ανάλογα με τον τύπο, στο πλαίσιο του αρχείου, εμφανίζεται το ανάλογο εικονίδιο του υποδεικνύει τον τύπο του.

Αναζήτηση αρχείων

Ταυτόχρονα, στην αρχική σελίδα του Drive, αλλά και μέσα σε κάθε φάκελο, παρέχεται η δυνατότητα αναζήτησης αρχείων, με κριτήριο το όνομα τους, και σε περίπτωση εισαγωγής γραμμάτων, στο κατάλληλο πεδίο αναζήτησης, εμφανίζονται μόνο τα σχετικά αποτελέσματα, των οποίων τα ονόματα περιέχουν τα γράμματα που εισήχθησαν.

Registry

Εναλλακτικά, με την επιλογή του Registry, ο χρήστης πλοηγείται στη σελίδα προβολής και εισαγωγής μητρώων ασθενών. Κατά την πρώτη είσοδο στην εφαρμογή, ο χρήστης δεν βλέπει καθόλου στοιχεία, καθώς δεν υπάρχουν καταχωρήσεις. Σε περίπτωση προηγούμενων καταχωρήσεων, ο χρήστης δύναται να πατήσει το κουμπί προβολής, όπου σε αυτή τη περίπτωση εμφανίζονται όλες οι καταχωρήσεις, σε μορφή πίνακα.

Προσθήκη καταχωρήσεων

Σε περίπτωση που ο χρήστης επιθυμεί, προσθέτει μια νέα καταχώρηση, πατώντας το αντίστοιχο κουμπί προσθήκης. Μόλις το επιλέξει, εμφανίζεται στην οθόνη μια αναδυόμενη φόρμα, όπου ο χρήστης πρέπει να καταχωρήσει τα κατάλληλα στοιχεία, στα αντίστοιχα πεδία. Υπάρχουν συνολικά δέκα πεδία φόρμας προς συμπλήρωση, με τα 8 να είναι υποχρεωτικά, ενώ σε περίπτωση λανθασμένης τιμής (π χ γράμμα αντί για αριθμό), ο χρήστης δεν μπορεί να ολοκληρώσει την διαδικασία καταχώρησης, έως ότου συμπληρωθούν σωστά τα υποχρεωτικά στοιχεία. Τα πεδία εισαγωγής έχουν εξ αρχής κόκκινο χρώμα, ενώ σε περίπτωση εισαγωγής αποδεκτών στοιχείων, μεταβάλλονται σε πράσινο. Ο χρήστης μπορεί να ακυρώσει την

υποβολή της φόρμας, είτε επιλέγοντας το κατάλληλο κουμπί, είτε πατώντας “κλικ” έξω από το παράθυρο της φόρμας, όπου σε αυτές τις περιπτώσεις η φόρμα καθίσταται κρυμμένη.

Όνοματεπώνυμο	
Ηλικία	
Βάρος	
Ύψος	

Επεξεργασία καταχωρήσεων

Σε περίπτωση λανθασμένης καταχώρησης ή επιθυμίας ανανέωσης, ο χρήστης, με δύο διαδοχικών “κλικ” μπορεί να επεξεργαστεί κάθε πεδίο του πίνακα ξεχωριστά και στη συνέχεια, πατώντας Enter, μπορεί να αποθηκεύσει την αλλαγή.

Εξαγωγή στοιχείων

Η εφαρμογή παρέχει τη δυνατότητα εξαγωγής του συνόλου του μητρώου ασθενή σε αρχείο csv, αφαιρώντας αναγνωριστικά στοιχεία ασθενών, όπως ακριβώς προβλέπεται από τη νομοθεσία της Ευρωπαϊκής Ένωσης. Τα στοιχεία δύναται να αξιοποιηθούν από κατάλληλους ιατρικούς οργανισμούς, με σκοπό την πρόοδο της έρευνας πάνω στην ασθένεια του διαβήτη.

Αποθήκευση Αρχείων

Για την αποθήκευση όλων των δεδομένων της εφαρμογής, αξιοποιείται το διαδικτυακό περιβάλλον ανάπτυξης κώδικα Firebase. Επιλέξαμε την συγκεκριμένη υλοποίηση καθώς προσφέρει σημαντικά εργαλεία, τα οποία απλοποιούν, έως εξαλείφουν, την ανάγκη παραμετροποίησης εξυπηρετητή (server) και βάσης δεδομένων, προσφέροντας μια έτοιμη, αλλά ταυτόχρονα συνεπή και ασφαλή υλοποίηση.

Διαγράμματα

Στην σελίδα του Registry, υπάρχει κουμπί με την επιγραφή Statistics, το οποίο μετά την επιλογή του με δεξί “κλικ” μας μεταφέρει στη σελίδα προβολής Διαγραμμάτων. Για όλα τα στοιχεία που έχουν καταχωρηθεί στο registry, δημιουργούνται ξεχωριστοί πίνακες σε μορφή πίτας (pie charts), οι οποίοι επιδεικνύουν την κατανομή των καταχωρήσεων ανά κατηγορία. Στην ίδια σελίδα υπάρχει κουμπί με την επιγραφή “Apply filters”, όπου όταν επιλεγθεί αναδύεται ένα παράθυρο μέσα στο οποίο ο χρήστης μπορεί να θέσει όρια για κάθε μια από τις κατηγορίες, με την αλλαγή στα διαγράμματα να γίνεται δυναμικά. Παράλληλα, παρέχονται επιπλέον δύο επιπλέον κουμπιά: το κουμπί κλεισίματος του παραθύρου και το κουμπί με την επιγραφή ‘>=’ το οποίο αλλάζει το όριο να είναι μεγαλύτερο ίσο ή μικρότερο ίσο από την επιλογή που θα εισάγει ο χρήστης. Σε περίπτωση επιλογής του κουμπιού, η επιγραφή του αλλάζει σε ‘<=’. Τέλος, παρέχεται κουμπί για την αφαίρεση όλων των φίλτρων που έχουν εφαρμοστεί πάνω στα διαγράμματα.

B.2 Πλεονεκτήματα εφαρμογής

Προσβασιμότητα στα δεδομένα

Η προτεινόμενη εφαρμογή απλοποιεί την αποθήκευση και διαχείριση των εισαγόμενων δεδομένων, γεγονός που επιτυγχάνεται μέσω της απλοποιημένης και κεντροποιημένης διεπαφής χρήστη. Η εγγραφή και είσοδος στο σύστημα, με αξιοποίηση συνδυασμού email και password, αποτελεί μια από τις συνηθέστερες μεθόδους που αξιοποιούνται σε ιστοσελίδες ή διαδικτυακές εφαρμογές.

Προαγωγή συνεργασίας και έρευνας

Η εν λόγω εφαρμογή, προάγει τη συνεργασία μεταξύ ασθενών, ιατρικού προσωπικού και οργανισμών, καθώς διαθέτει τη δυνατότητα εξαγωγής και διαμοιρασμού των δεδομένων. Ταυτόχρονα όλα τα εξαγόμενα δεδομένα δύναται να χρησιμοποιηθούν για ερευνητικούς σκοπούς από ιατρικούς οργανισμούς αλλά και αυτόνομους επαγγελματίες στο χώρο της υγείας, συνεισφέροντας στην περαιτέρω μελέτη και αντιμετώπιση της ασθένειας.

Ασφάλεια και Ιδιωτικότητα

Η εφαρμογή σχεδιάστηκε έχοντας κύριο γνώμονα την ασφάλεια και την ακεραιότητα των δεδομένων που εισάγονται και αποθηκεύονται. Με την υλοποίηση και παραμετροποίηση του Firebase, ουσιαστικά υιοθετούμε τα πρότυπα ασφαλείας που παρέχει η πλατφόρμα. Πιο συγκεκριμένα, παρέχοντας ειδικές οδηγίες στο Firebase, έχουμε τη δυνατότητα να υλοποιήσουμε ένα access control protocol, όπου μόνο οι πιστοποιημένοι και συνδεδεμένοι χρήστες έχουν την δυνατότητα να προβούν σε προβολή (Read) και εγγραφή (Write) στην βάση δεδομένων που παρέχεται από το Firebase.

Συμμόρφωση με GDPR

Παράλληλα, η εφαρμογή είναι (σχεδόν) GDPR Compliant, καθώς τηρούνται όλα τα δικαιώματα χρήστη που παρουσιάζονται παρακάτω:

1. Δικαίωμα στη Πληροφορία και Διαφάνεια
2. Δικαίωμα στην Πρόσβαση
3. Δικαίωμα στην Επικαιροποίηση
4. Δικαίωμα στην Διαγραφή
5. Δικαίωμα σε Άρνησης Επεξεργασίας Δεδομένων
6. Δικαίωμα Φορητότητας
7. Δικαίωμα σε Ένσταση
8. Δικαίωμα Άρνησης σε Αυτοματοποιημένης Λήψεις Αποφάσεων

B.3 Προϋπάρχουσες Υλοποιήσεις και Μελέτες

Κατά την έρευνα μας σε διάφορες πηγές για σχετική βιβλιογραφία, συναντήσαμε ενδιαφέρουσες μελέτες αλλά και υλοποιήσεις που αφορούν καταγραφή και αποθήκευση δεδομένων ασθενών, αξιοποιώντας ψηφιακά αρχεία υγείας.

Development of a diabetes registry to improve quality of care in the National Healthcare Group in Singapore

Η παραπάνω δημοσίευση παρουσιάζει την προσπάθεια ανάπτυξης συστήματος καταγραφής μητρώων για το Εθνικό Ινστιτούτο Υγείας της Σιγκαπούρης^[10]. Παρ' όλο που η μελέτη δημοσιεύθηκε το 2009, το άρθρο αναφέρει ότι υπήρξαν πρώιμες έρευνες σχετικά με την ανάπτυξη του εν λόγω συστήματος. Το προτεινόμενο σύστημα παρέχει δυνατότητες αποθήκευσης και προβολής των ιατρικών αρχείων ασθενών από ιατρούς, σε συνδυασμό με πληθώρα γραφημάτων που απεικονίζουν τη πορεία του κάθε ασθενούς. Η κάθε καρτέλα με τα δεδομένα του ασθενή συνοδεύεται με τις ιατρικές αποφάσεις του εκάστοτε θεράποντος επαγγελματία, υποστηριζόμενες από εργαστηριακά ευρήματα.

Τέλος, η έρευνα καταλήγει στο συμπέρασμα ότι μετά την συστηματική παρακολούθηση και εξατομικευμένη διαχείριση κάθε ασθενούς, παρατηρήθηκαν βελτιώσεις στην ποιότητα ζωής αλλά και στους βιοδείκτες των ασθενών.

eHealth technologies to support nutrition and physical activity behaviours in diabetes self-management

Η παραπάνω μελέτη επικεντρώνεται σε διατροφικά δεδομένα και δεδομένα που προκύπτουν κατά την άσκηση, και πώς αυτά μπορούν να συνεισφέρουν στην διαχείριση του διαβήτη^[11]. Παρέχεται ανασκόπηση σε τεχνολογίες που αξιοποιούνται για την καταγραφή των δεδομένων, όπως κινητές συσκευές, wearables αλλά και video games που βασίζονται στην άσκηση του χρήστη. Τέλος, παρατίθενται τομείς πάνω στους οποίους πρέπει να βασιστούν μελλοντικές έρευνες.

Overlap

Το overlap αποτελεί αυτοτελή πλατφόρμα η οποία μπορεί να υιοθετηθεί από ιατρικούς οργανισμούς, και να προσφέρει αποθήκευση, καταγραφή και οπτικοποίηση ιατρικών δεδομένων^[12]. Ο εκάστοτε ιατρικός οργανισμός σχεδιάζει τη δική του εφαρμογή βασισμένη σε προϋπάρχουσες βιβλιοθήκες του overlap, συνδέει την εφαρμογή με το παράθυρο

διαχείρισης (control panel) και στη συνέχεια είναι σε θέση να λαμβάνει δεδομένα ανά πάσα στιγμή, σε ζωντανό χρόνο.

Vivify Health

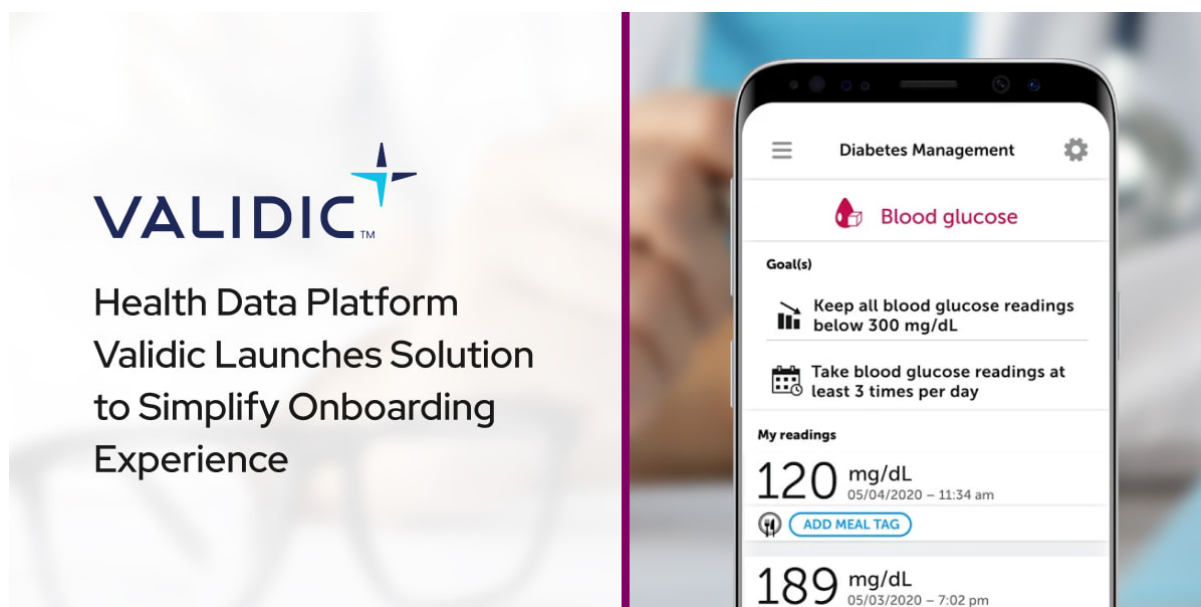
Το Vivify Health αποτελεί επίσης πλατφόρμα για τη διαχείριση ιατρικών δεδομένων^[13]. Η λειτουργικότητα της μοιάζει αρκετά με το overlap, με τις κύριες διαφορές να είναι ότι το vivify επικεντρώνεται στην επικοινωνία ιατρού με ιατρό (Clinician to Clinician), με την αξιοποίηση εφαρμογών και wearables. Παρουσιάζει ενδιαφέρουσες λειτουργικότητες, όπως σύνδεση με ΑΙ μέσω Προγραμματιστικής διεπαφής χρήστη (Application Programming Interface), αλλά και έλεγχο της εφαρμογής αξιοποιώντας φωνητικές εντολές.



Εικόνα 2: Η αρχική οθόνη της εφαρμογής «Vivify Health»

Validic

Το Validic αποτελεί ολοκληρωμένη πλατφόρμας καταγραφής και αποθήκευσης ιατρικών δεδομένων^[14]. Η υπηρεσία λειτουργεί επίσης όπως οι 2 προαναφερθείσες πλατφόρμες, με την αξιοποίηση εφαρμογών και wearables για τη λήψη των δεδομένων. Η πλατφόρμα προορίζεται για χρήση από ασθενείς και ιατρούς, ενώ παρέχει χρήσιμα εργαλεία για την οπτικοποίηση αλλά και την εξαγωγή συμπερασμάτων από τα δεδομένα. Η εφαρμογή έχει “χτιστεί” έτσι ώστε να είναι κλιμακωτή (scalable), δηλαδή να μπορεί να υιοθετηθεί από μικρούς αλλά και μεγαλύτερους οργανισμούς.



Εικόνα 3: Η αρχική οθόνη της εφαρμογής «Validic»

Γ. ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΑΞΙΟΠΟΙΗΘΗΚΑΝ

Javascript

Η Javascript αποτελεί το κύριο εργαλείο που χρησιμοποιήθηκε για την εκπόνηση της εφαρμογής. Αποτελεί γλώσσα προγραμματισμού υψηλού επιπέδου, η οποία αξιοποιείται από τη πλειοψηφία των διαδικτυακών ιστοσελίδων, με σκοπό την προβολή διαδραστικού και δυναμικού περιεχομένου. Υποστηρίζεται από όλους τους μοντέρνους Περιηγητές (browsers), γεγονός που την καθιστά αξιόπιστη επιλογή για ανάπτυξη ιστοσελίδων. Η γλώσσα, κυρίως, εκτελείται στην πλευρά του χρήστη (client side), ενώ με την προσθήκη Πλαισίων Ανάπτυξης Εφαρμογών (frameworks), όπως το Node.js, μπορεί να εκτελεστεί και στην πλευρά του διακομιστή (server side)^[15].

React

Η React αποτελεί μια από τις δημοφιλέστερες βιβλιοθήκες Javascript και αξιοποιείται για την δημιουργία διεπαφών χρήστη (User Interfaces), οι οποίες προορίζονται για χρήση από διαδικτυακές εφαρμογές ή σελίδες. Η React αναπτύσσεται και συντηρείται από την εταιρεία Meta (πρώην Facebook). Ιδιαίτερο χαρακτηριστικό της αποτελεί το γεγονός ότι η δομή της επιτρέπει την συγγραφή κώδικα JS, HTML και CSS σε ένα και μόνο αρχείο, καθιστώντας την αρκετά ευέλικτη. Κάθε ένα τέτοιο αρχείο αποτελεί ένα συστατικό (component) της εφαρμογής και αντιπροσωπεύουν όλα τα επιμέρους κομμάτια της ^[16].

HTML

Η HTML (Hypertext Markup Language) αποτελεί γλώσσα markup και αξιοποιείται για την ορισμό της δομής και του περιεχομένου διαδικτυακών ιστοσελίδων. Με τη χρήση προκαθορισμένων tag και elements, μπορούμε να ορίσουμε τη σχέση μεταξύ διαφόρων συστατικών (components) της εφαρμογής ^[17].

CSS

Η CSS (Cascading Style Sheets) αποτελεί γλώσσα προγραμματισμού, η οποία χρησιμοποιείται για την διαμόρφωση των HTML αρχείων. Με τη προσθήκη ενός ονόματος κλάσης (className) σε κάποιο στοιχείο (element), μπορούμε να προσθέσουμε χαρακτηριστικά, όπως το μέγεθος, το χρώμα και άλλα, που αφορούν την παρουσίαση (presentation) του εκάστοτε element [\[18\]](#).

Node.js

Το Node.js αποτελεί περιβάλλον εκτέλεσης κώδικα και βασίζεται στο V8 JS Engine του Google Chrome, και καθιστά δυνατή την εκτέλεση κώδικα Javascript στη πλευρά του εξυπηρετητή (server). Το Node παρέχει έτοιμα modules, τα οποία αξιοποιούνται για διαχείριση αρχείων συστήματος, εκτέλεση αιτημάτων HTTP. Το περιβάλλον παρέχει επίσης βιβλιοθήκες από ξένες οντότητες (third-party libraries), οι οποίες γίνονται διαθέσιμες με χρήση του λογισμικού Διαχείρισης Πακέτων Node (Node Package Manager).

Συμπερασματικά, με τον συνδυασμό του Node με κάποιο JS framework όπως React ή Angular είμαστε σε θέση να δημιουργήσουμε πλήρεις εφαρμογές (full stack applications) [\[19\]](#).

Firebase

Το Firebase αποτελεί διαδικτυακή πλατφόρμα ανάπτυξης κώδικα και διατίθεται από τη google [\[20\]](#). Παρέχει ποικιλία εργαλείων και υπηρεσιών που σχετίζονται με την ανάπτυξη (develop), παράταξη (deploy) και διαχείριση διαδικτυακών εφαρμογών.

Το Firebase εξαλείφει την ανάγκη ύπαρξης δομής διαχείρισης εξυπηρετητή, επιτρέποντας στους developers να επικεντρώνονται στις καίριες λειτουργίες της εφαρμογής τους. Στη περίπτωση μας, το Firebase αξιοποιείται για την αυθεντικοποίηση των χρηστών και την αποθήκευση των δεδομένων της εφαρμογής.

Vite

Το Vite αποτελεί εργαλείο κατασκευής (build) και εξυπηρετητή ανάπτυξης λογισμικού (development server). Επιτρέπει τη δημιουργία project, τα οποία είναι πλήρως συμβατά με όλες τις προαναφερθείσες τεχνολογίες, ενώ ταυτόχρονα βελτιστοποιεί την ταχύτητα με την οποία διαβάζεται και εκτελείται ο κώδικας από τον εξυπηρετητή^[21].

Node Package Manager

Το npm αποτελεί λογισμικό διαχείρισης πακέτων του Node.js για τη Javascript και αξιοποιήθηκε για την λήψη και εγκατάσταση του συνόλου των εργαλείων που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής^[22].

Three.js

Η three.js αποτελεί library της javascript και αξιοποιείται για δημιουργία και προβολή τρισδιάστατων γραφικών (computer graphics)^[23]. Αποτελείται από τη σκηνή, τα αντικείμενα, τη κάμερα και το renderer, τα οποία σαν σύνολο δημιουργούν ένα περιβάλλον που μπορούμε να δημιουργήσουμε και να προβάλλουμε τρισδιάστατα αντικείμενα.

Framer-Motion

Το Framer-Motion αποτελεί βιβλιοθήκη ανοιχτού περιεχομένου (open source) και χρησιμοποιείται για την δημιουργία διαδραστικών σχεδίων (animations). Στη περίπτωση της εφαρμογής μας, αξιοποιείται στο background της σελίδας Εγγραφής και Εισόδου, καθώς και στη σελίδα υποδοχής όπου διατίθενται δύο διαδραστικά σχήματα που χρησιμοποιούνται από το χρήστη για πλοήγηση^[24].

Tailwind

Το Tailwind είναι framework της css και προσφέρει εξατομικευμένες επιλογές αναφορικά με την εμφάνιση και μορφοποίηση των HTML elements ^[25]. Αυτό επιτυγχάνεται αξιοποιώντας ειδικά ονόματα κλάσεων, τα οποία προστίθενται στο όνομα της κλάσης του HTML element, έτσι ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα.

Bootstrap

Παράλληλα με το tailwind, αξιοποιήθηκε ένα επιπλέον css framework, το Bootstrap^[26]. Το Bootstrap επίσης λειτουργεί με προσθήκες ειδικών ονομάτων κλάσεων, καθιστώντας τα HTML elements διαδραστικά και κλιμακωτά (scalable).

Charts.js

Για την υλοποίηση των διαγραμμάτων επιλέχθηκε η βιβλιοθήκη Charts.js, η οποία αποτελεί βιβλιοθήκη που παρέχει δυνατότητες προβολής διαγραμμάτων, αλλά και συναρτήσεις και εργαλεία για την προβολή και παραμετροποίηση τους^[27].

Δ. ΟΘΟΝΕΣ

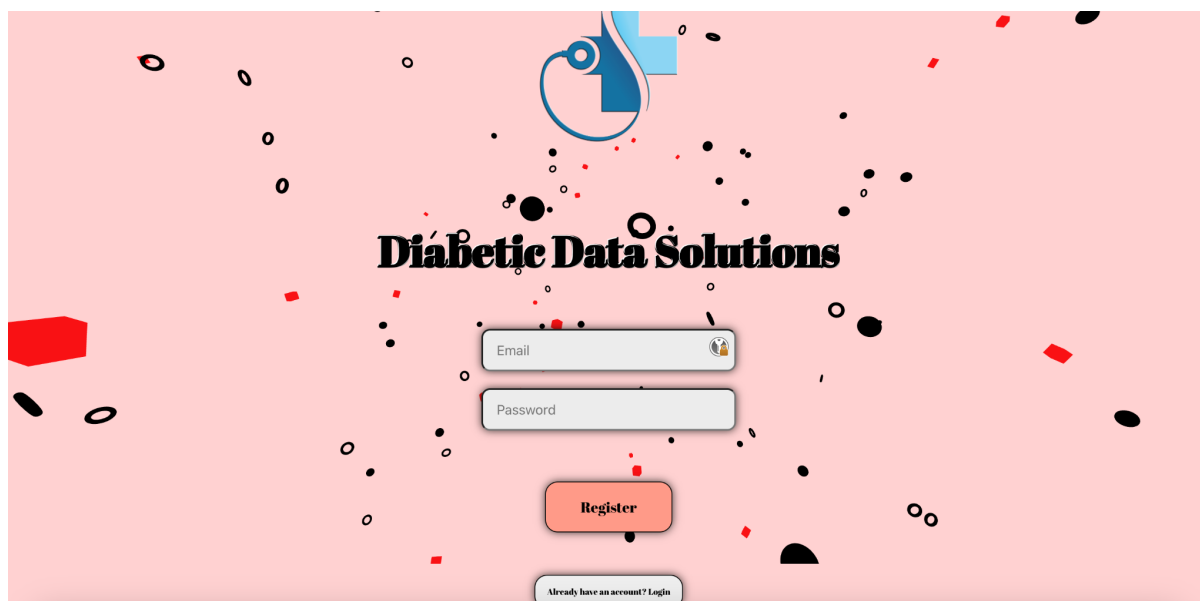
Δ.1. Οθόνες Εφαρμογής

Η πρώτη σελίδα που συναντά ο χρήστης είναι η σελίδα Υποδοχής στην οποία είτε εισάγονται τα στοιχεία πρόσβασης και ο χρήστης επιλέγει “Login” (Είσοδος), είτε ο χρήστης μπορεί να επιλέξει το κουμπί εγγραφής.



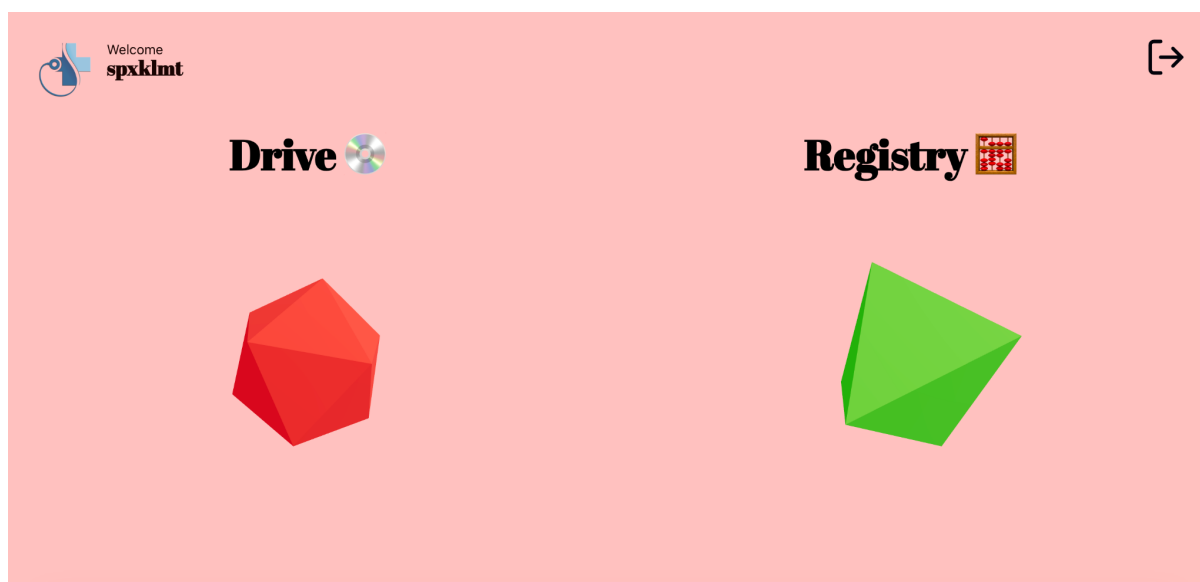
Εικόνα 4: Οθόνη εισόδου στην εφαρμογή

Σε περίπτωση που ο χρήστης επιλέξει το κουμπί εγγραφής, τότε ανακατευθύνεται στη σελίδα εγγραφής, όπου εισάγει τη διεύθυνση ηλεκτρονικού ταχυδρομείου του και τον κωδικό που επιθυμεί και επιλέγει το κουμπί “εγγραφή”. Ακριβώς κάτω από το συγκεκριμένο κουμπί, υπάρχει ένα άλλο κουμπί που τον δρομολογεί στην σελίδα σύνδεσης.



Εικόνα 5: Οθόνη εγγραφής στην εφαρμογή

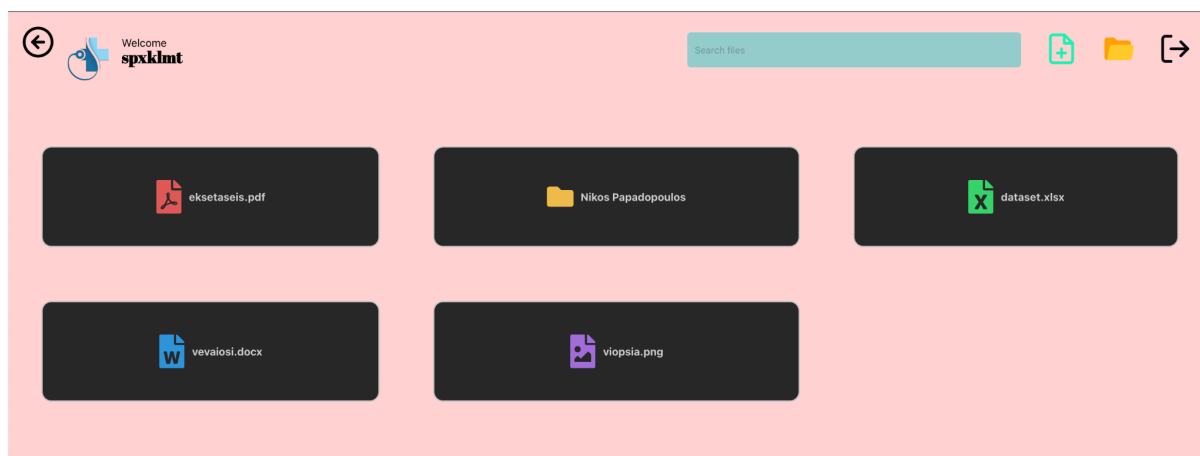
Αφού ο χρήστης συνδεθεί, δρομολογείται στην σελίδα, όπου μπορεί να επιλέξει ανάμεσα σε πρόσβαση στον χώρο αποθήκευσης (Drive), ή στο αρχείο καταγραφών (Registry). Επίσης, μπορεί να επιλέξει το κουμπί πάνω δεξιά με το οποίο μπορεί να αποσυνδεθεί από την εφαρμογή.



Εικόνα 6: Αρχική οθόνη εφαρμογής

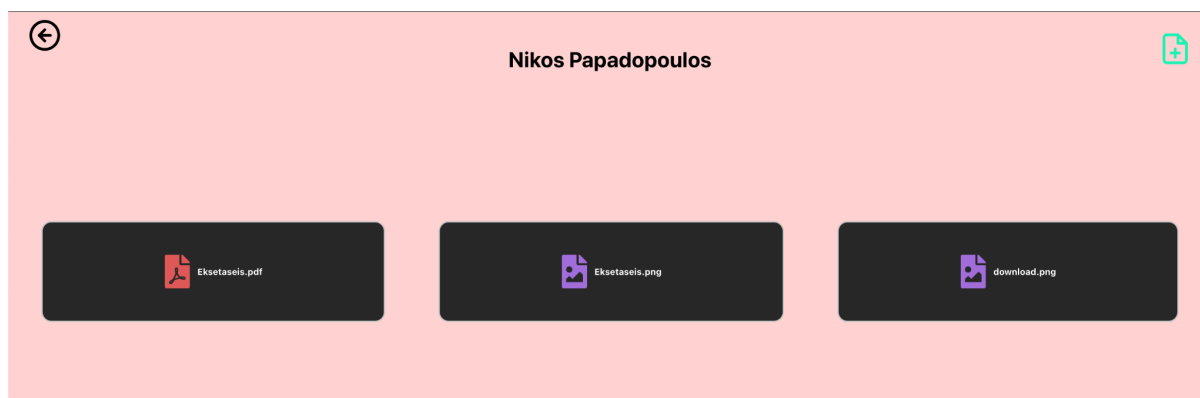
Εάν ο χρήστης επιλέξει την πρόσβαση στον χώρο αποθήκευσης (Drive), δρομολογείται στην αρχική σελίδα του χώρου αποθήκευσης, όπου μπορεί να μεταφορτώσει ένα νέο αρχείο, να δημιουργήσει έναν νέο φάκελο, να ανοίξει ένα αρχείο ή ένα φάκελο και να προβάλει ή να

κατεβάσει ένα ή περισσότερα αρχεία. Επίσης, με δεξί “κλικ” πάνω στον φάκελο ή το αρχείο, ο χρήστης μπορεί να τα μετονομάσει ή να τα διαγράψει.



Εικόνα 7: Αρχική οθόνη χώρου αποθήκευσης (drive)

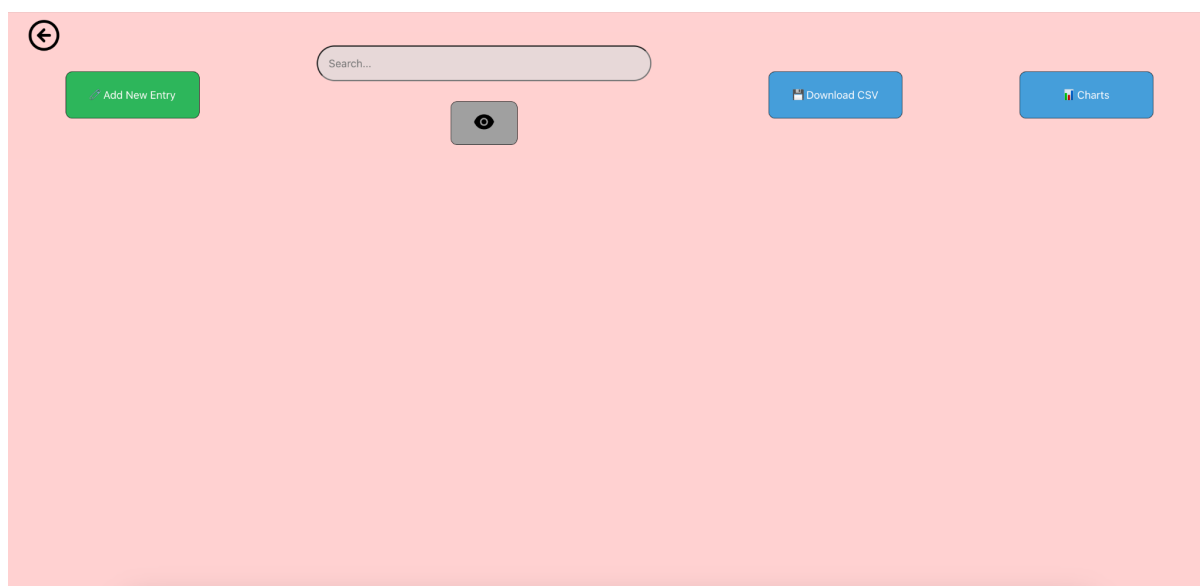
Σε περίπτωση που ο χρήστης επιλέξει να εισέλθει σε έναν φάκελο, στο πάνω μέρος της οθόνης εκτυπώνεται το όνομα του φακέλου καθώς και τα περιεχόμενα του. Υπάρχουν οι λειτουργίες της προσθήκης ενός νέου αρχείου μέσω μεταφόρτωσης, της επιστροφής στην αρχική σελίδα του χώρου αποθήκευσης, καθώς και η μετονομασία ή διαγραφή των υπαρχόντων αρχείων με δεξί κλικ πάνω στο αρχείο.



Εικόνα 7: Οθόνη εντός φακέλου στον χώρο αποθήκευσης

Σε περίπτωση που ο χρήστης επιλέξει να εισέλθει στην κατηγορία Registry θα δρομολογηθεί σε αυτή τη σελίδα όπου ο χρήστης μπορεί να προσθέσει μία νέα εγγραφή, να αναζητήσει εγγραφές, να προβάλλει στην οθόνη του όλες τις υπάρχουσες εγγραφές και να εξαγάγει όλες τις υπάρχουσες εγγραφές χωρίς ονοματεπώνυμα. Μία ακόμα λειτουργία που παρέχεται, είναι η προβολή όλων των δεδομένων σε διαγράμματα. Οι ήδη υπάρχουσες

καταγραφές δεν εμφανίζονται στην οθόνη κατά την πρώτη είσοδο, ενώ για την εμφάνιση τους ο χρήστης οφείλει να επιλέξει το γκρι κουμπί κάτω από τη μπάρα αναζήτησης.



Εικόνα 8: Αρχική οθόνη μητρώου

Προβολή καταχωρήσεων/εγγραφών:

Full Name	Age	Weight	Height	Gender	Date	Glucose After Fasting	HbA1c	Glucose from Mouth 1h	Glucose from Mouth 2h
Νικόλαος Παπαδοπούλος	57	80	170	Male	20/04/2023	115	3	160	120
Γεώργιος Νικολοπούλος	37	65	178	Male	22/04/2023	125	7	170	140
Αννα Γιαννοπούλου	40	70	168	Female	25/05/2023	110	4	150	120
Μαρία Πουλοπούλου	30	80	170	Female	20/06/2023	130	9	190	150

Εικόνα 9: Οθόνη μητρώου μετά από εμφάνιση καταχωρήσεων

Εάν ο χρήστης επιλέξει να προσθέσει μία νέα εγγραφή στο σύστημα, θα αναδυθεί παράθυρο όπου θα πρέπει να εισάγει τα υποχρεωτικά στοιχεία που εμφανίζονται με κόκκινο, ακριβώς όπως εμφανίζεται στη παρακάτω εικόνα.

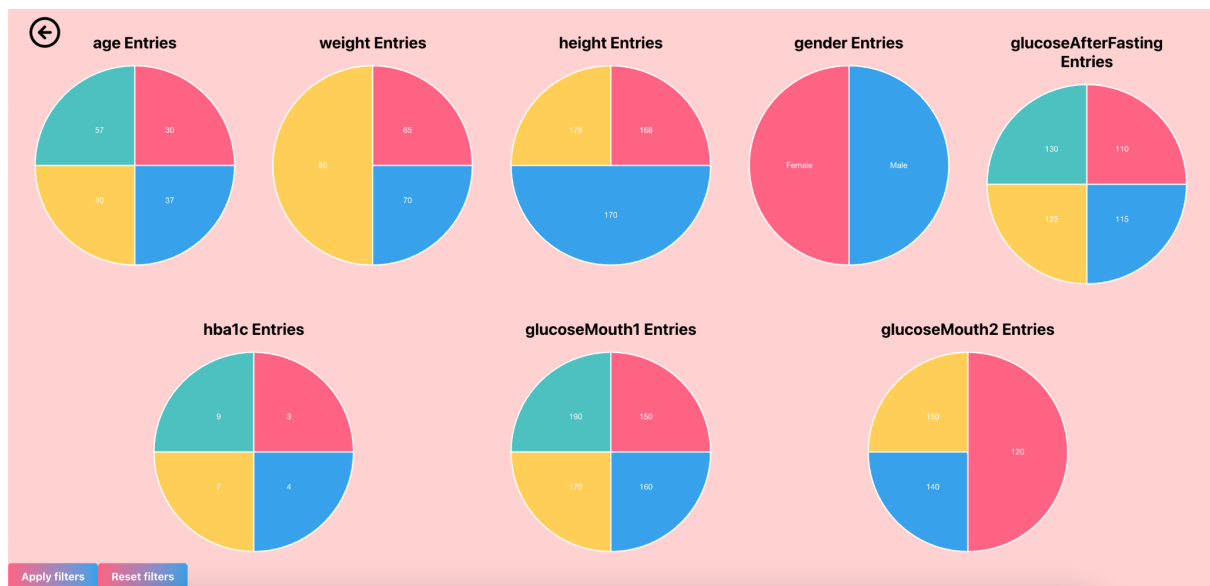
Εικόνα 10: Οθόνη νέας καταχώρησης

Επίσης, ο χρήστης μπορεί να προβεί σε επεξεργασία καταχώρησης, όταν μετά από επιλογή ή αναζήτηση καταχωρήσεων κάνει διπλό “κλικ” πάνω στο πεδίο του της εκάστοτε καταχώρησης.

Full Name	Age	Weight	Height	Gender	Date	Glucose After Fasting	HbA1c	Glucose from Mouth 1h	Glucose from Mouth 2h
Νικόλαος Παπαδόπουλος	57	80	170	Male	20/04/2023	115	3	160	120
Γεωργιος Νικολαοπουλος	37	65	178	Male	22/04/2023	125	7	170	140
Αννα Γιαννοπουλοῦ	40	70	168	Female	25/05/2023	110	4	150	120
Μαρια Πουλοπουλου	30	80	170	Female	20/06/2023	130	9	190	150

Εικόνα 11: Οθόνη επεξεργασίας στοιχείων καταχώρησης

Εάν η ο χρήστης επιλέξει να ανοίξει την σελίδα των διαγραμμάτων, θα ανακατευθυνθεί στην παρακάτω οθόνη όπου μπορεί να δει τα διαγράμματα όλων των τιμών που έχουν εισαχθεί στο σύστημα καθώς και να επιστρέψει στην αρχική οθόνη του μητρώου.



Εικόνα 12: Οθόνη διαγραμμάτων

Επίσης, ο χρήστης μπορεί να προσθέσει φίλτρα στα εμφανιζόμενα διαγράμματα, όπως και να αφαιρέσει όλα τα φίλτρα που έχει προσθέσει.

Εάν επιλέξει να προσθέσει φίλτρα, θα εμφανιστεί το παρακάτω αναδυόμενο παράθυρο που δίνει στον χρήστη την δυνατότητα να επιλέξει τι θέλει να βλέπει στα διαγράμματα. Στο κάτω μέρος του παραθύρου με ένα κλικ στο σύμβολο “>=” μπορεί να το αλλάξει σε “<=” εάν επιθυμεί να αλλάξει την ρύθμιση του φίλτρου.

Age Threshold:

Weight Threshold:

Height Threshold:

GlucoseAfterFasting Threshold:

Hba1c Threshold:

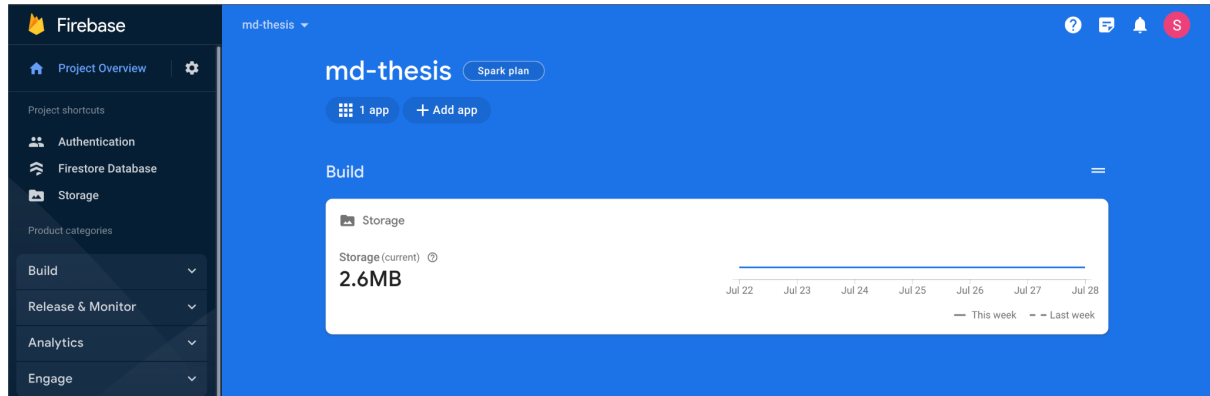
GlucoseMouth1 Threshold:

GlucoseMouth2 Threshold:

Εικόνα 13: Οθόνη εφαρμογής φίλτρων στα διαγράμματα

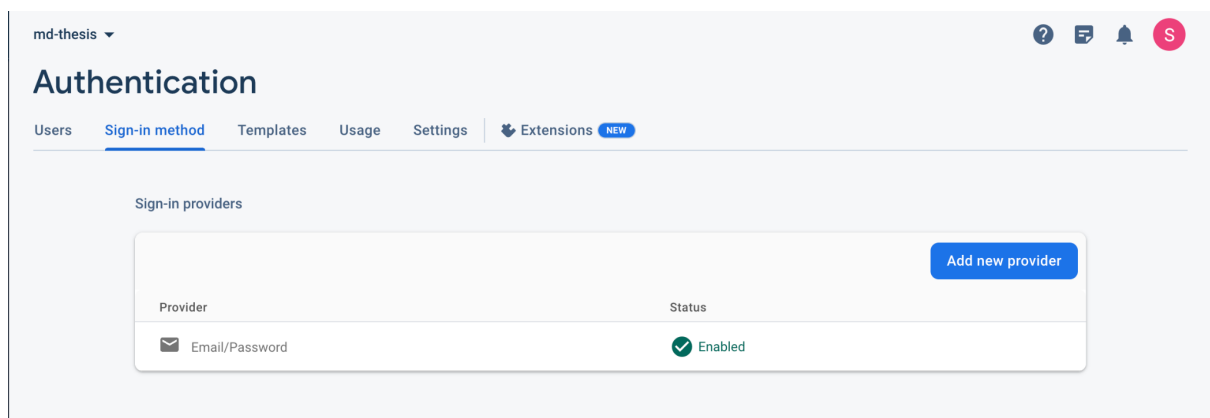
Δ.2 Οθόνες Firebase

Στην αρχική οθόνη του Firebase βλέπουμε χρήσιμα δεδομένα και μπορούμε να πλοηγηθούμε στις υποκατηγορίες που ακολουθούν:



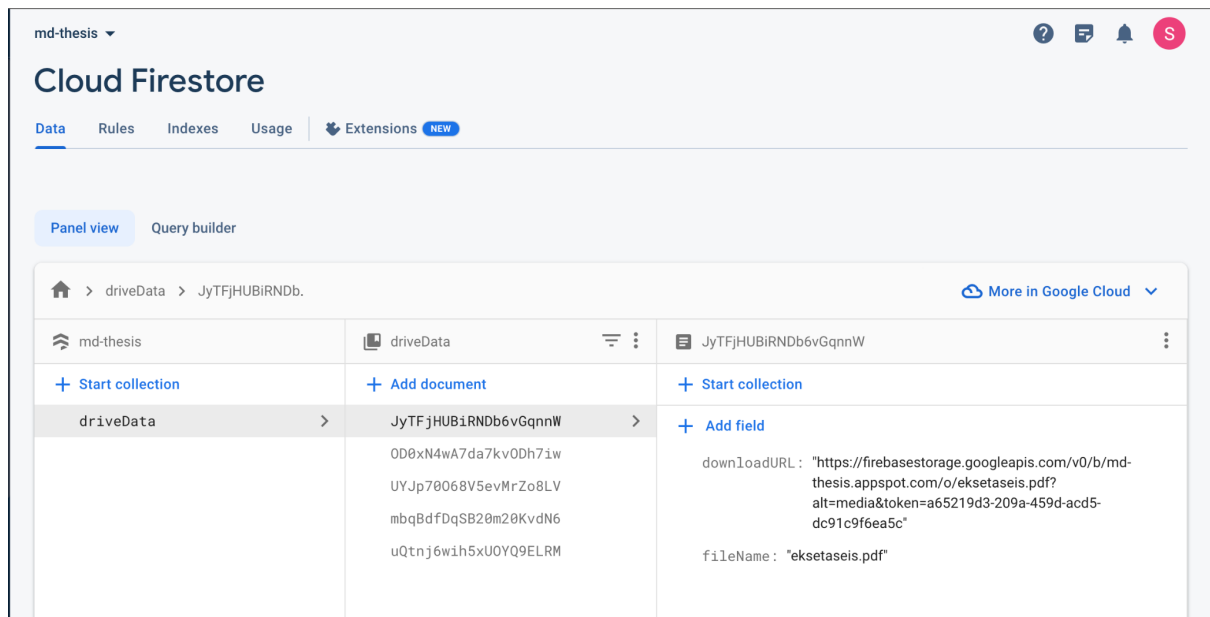
Εικόνα 14 Αρχική Οθόνη Firebase

Οθόνη ρύθμισης μεθόδου αυθεντικοποίησης στο σύστημα μέσω Firebase, όπου παρουσιάζεται η μέθοδος σύνδεσης στην εφαρμογή



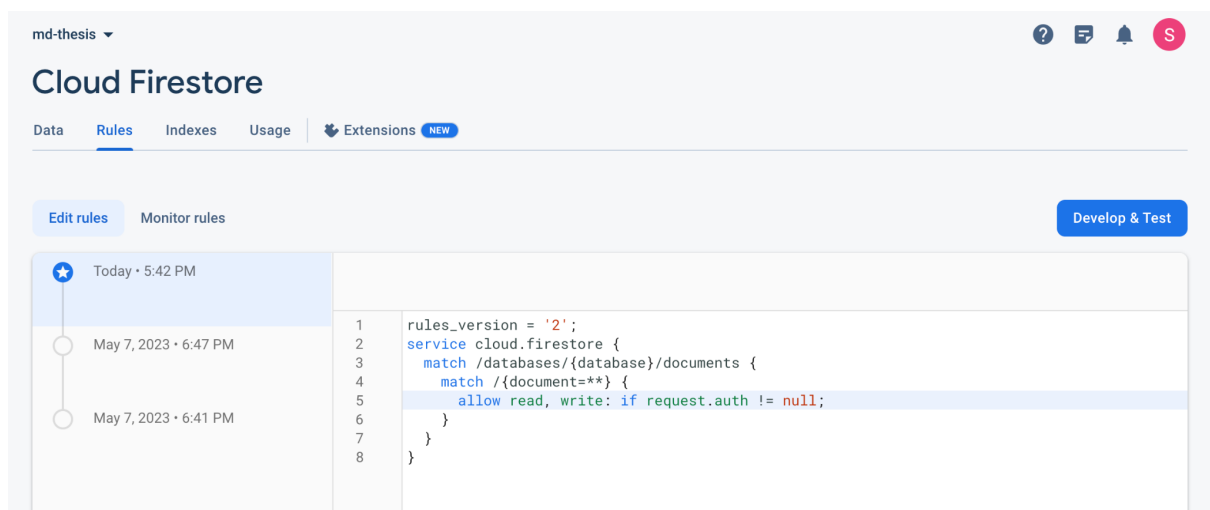
Εικόνα 15: Οθόνη ρύθμισης μεθόδου αυθεντικοποίησης

Οθόνη προβολής των συλλογών που περιέχουν τον τίτλο ενός αρχείου κι έναν σύνδεσμο που αξιοποιείται για πρόσβαση λήψης των αρχείων που είναι αποθηκευμένα στο Firebase Storage



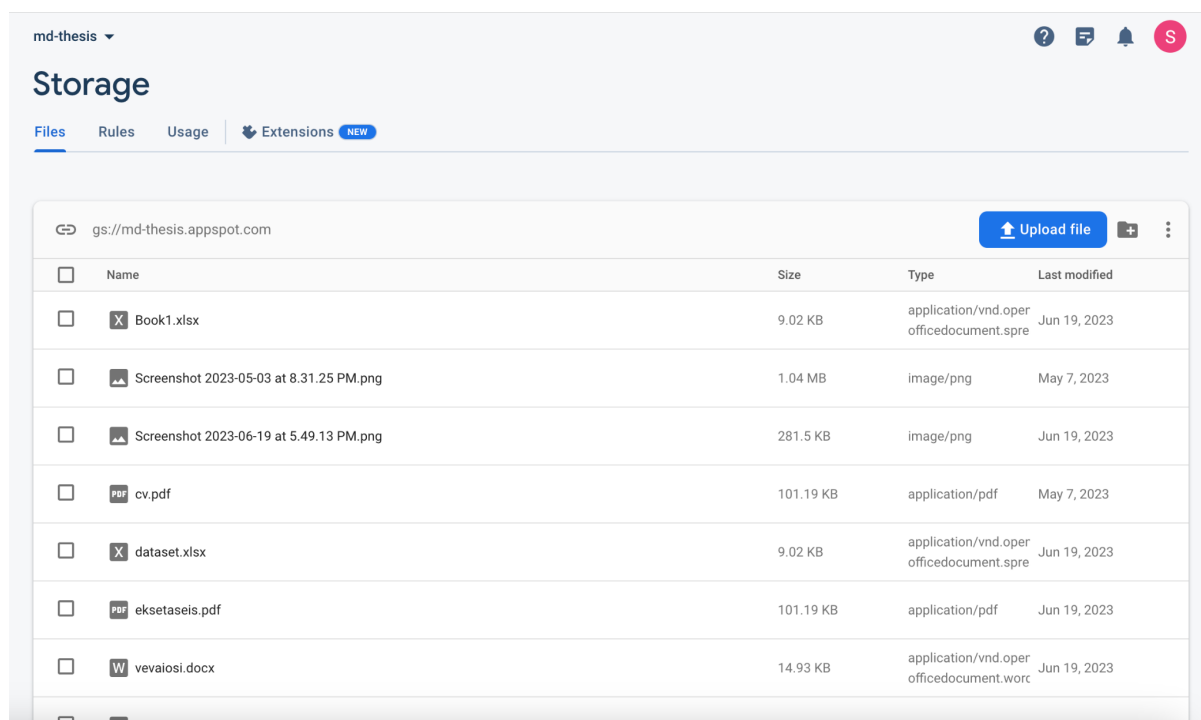
Εικόνα 16: Οθόνη δεδομένων του «Cloud Firestore»

Στην καρτελα «Rules» ρυθμίζουμε τους κανόνες που αφορούν σε ποιους επιτρέπεται η δημιουργία / επεξεργασία / διαγραφή στοιχείων από τη βάση δεδομένων της εφαρμογής



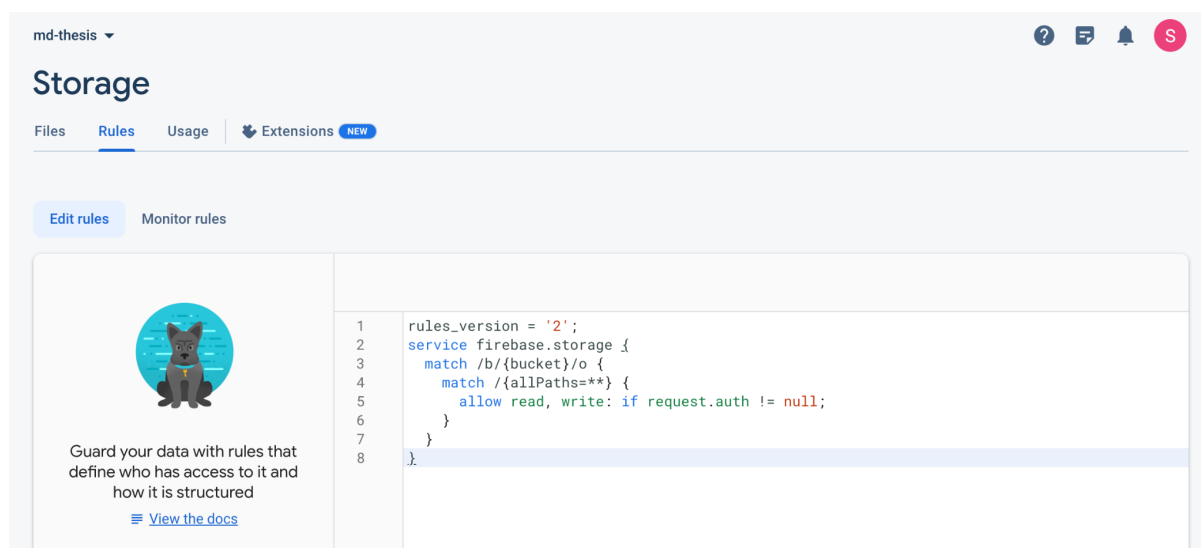
Εικόνα 17: Οθόνη Κανόνων που αφορούν τα δικαιώματα των χρηστών της εφαρμογής

Στην αρχική οθόνη της κατηγορίας «Storage» προβάλλονται τα αρχεία που βρίσκονται στον χώρο αποθήκευσης της εφαρμογής



Εικόνα 18: Οθόνη των αρχείων που βρίσκονται στον χώρο αποθήκευσης (Storage)

Στην καρτέλα «Rules» της κατηγορίας «Storage» υπάρχουν οι κανόνες για την πρόσβαση στα αρχεία που βρίσκονται στον χώρο αποθήκευσης



Εικόνα 19: Κανόνες που αφορούν την πρόσβαση των χρηστών στα αρχεία

Ε. ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ & ΣΥΜΠΕΡΑΣΜΑΤΑ

Ε.1 Προσθήκη 2FA για πρόσβαση στην εφαρμογή

Με στόχο την περαιτέρω προστασία των δεδομένων, ως μελλοντική προέκταση προτείνεται η χρήση αυθεντικοποίησης δύο παραγόντων (Two Factor Authenticator) μέσω της οποίας δημιουργείται ένα μοναδικό token ανά χρήστη, το οποίο παράγεται από εφαρμογές παραγωγής token, όπως το Google Authenticator της Google. Αυτή η εφαρμογή στην συνέχεια, ανά τριάντα δευτερόλεπτα, θα παράγει έναν εξαψήφιο κωδικό (token) μιας χρήσης, ο οποίος θα απαιτείται σε συνδυασμό με τον κωδικό του χρήστη κατά την διαδικασία αυθεντικοποίησης και εισόδου στην εφαρμογή.

Ε.2 Προσθήκη Αποκεντροποιημένης Βάσης Δεδομένων

Για την περαιτέρω εξασφάλιση και ενίσχυση της ασφάλειας των δεδομένων, προτείνουμε δημιουργία Αποκεντροποιημένης Βάσης Δεδομένων, η οποία δεν διαθέτει κάποιον διαχειριστή, αλλά δίνει πρόσβαση στα δεδομένα με βάση μόνο τους ρόλους και τα permissions του κάθε χρήστη. Η απουσία διαχειριστή επιτρέπει την εφαρμογή αυστηρότερων access protocols, ενώ ταυτόχρονα εξαλείφεται και ο ανθρώπινος παράγοντας από τη διαχείριση των δεδομένων, γεγονός που καθιστά τα δεδομένα λιγότερο ευάλωτα σε επιθέσεις phishing μέσω social engineering, αλλά και άλλες πολλές που βασίζονται κατ' εξοχήν στα ανθρώπινα λάθη.

Ε.3 Σύνδεση Εφαρμογής με Wearables

Ως μια εποικοδομητική μελλοντική προέκταση καταγράφεται η διασύνδεση της εφαρμογής με φορητές συσκευές (wearables), η οποία θα ανανεώνουν δυναμικά και σε ζωντανό χρόνο τη βάση δεδομένων με τους βιοδείκτες του χρήστη, γεγονός που έχει ως αποτέλεσμα ο χρήστης να ενημερώνεται τακτικά για τυχόν μεταβολές στους βιοδείκτες του. Με αυτό το τρόπο παρέχεται περαιτέρω ασφάλεια στους χρήστες που οφείλουν να είναι συνεχώς ενήμεροι για τις τιμές στο σώμα τους, αλλά και παρέχει περισσότερα και πολύ αναλυτικότερα δεδομένα προς ανάλυση.

Ε.4 Δημιουργία Πλατφόρμας Διαμοιρασμού Δεδομένων

Σε περίπτωση ευρείας υιοθέτησης της πλατφόρμας από ιατρικούς οργανισμούς, προδιαγράφεται η δημιουργία Πλατφόρμας Διαμοιρασμού Δεδομένων, όπου όλοι οι συμμετέχοντες οργανισμοί και επαγγελματίες θα μπορούν να διαμοιραστούν ανωνυμοποιημένα δεδομένα που συλλέχθηκαν με την ρητή συναίνεση των ασθενών. Αξίζει να αναφερθεί ότι σε αυτή τη περίπτωση επίσης υιοθετείται ένα κοινό πρότυπο (template) διατήρησης των αρχείων, γεγονός που συμβάλλει στην περαιτέρω βελτίωση των διαδικασιών διαμοιρασμού, ανάγνωσης και χρήσης των δεδομένων, ανάλογα με τη περίπτωση.

Ε.5 Δημιουργία Εφαρμογής για Ios, Android

Η παρούσα εφαρμογή έχει κατασκευαστεί ως μία διαδικτυακή εφαρμογή και όχι ως μία εφαρμογή για φορητές ηλεκτρονικές συσκευές (iOS/Android). Με αξιοποίηση της React Native, μπορούμε να προβούμε σε μετατροπή της διαδικτυακής εφαρμογής σε εφαρμογή για τις προαναφερθείσες συσκευές. Ταυτόχρονα, η εφαρμογή δύναται να συνδέεται με τα προαναφερθέντα wearables, παρέχοντας με ευκολία τα δεδομένα προς προβολή και χρήση για τον εκάστοτε χρήστη.

Ε.6 Υλοποίηση Φόρμας Παροχής Συναίνεσης για Επεξεργασία Δεδομένων

Σε συνδυασμό με την παραπάνω πρόταση οφείλουμε να προσθέσουμε μια παράλληλη σελίδα με Φόρμας Παροχής Συναίνεσης για Επεξεργασία Δεδομένων, όπου ο κάθε χρήστης θα έχει τη δυνατότητα να παράσχει συναίνεση για χρήση και/ή διαμοιρασμό συγκεκριμένων δεδομένων που ενδέχεται να διατηρούνται από συμμετέχοντες στο σύστημα ιατρικούς οργανισμούς ή επαγγελματίες. Ταυτόχρονα, με κάθε συνεισφορά δεδομένων και μετέπειτα χρήσης τους από οργανισμούς, ο χρήστης θα λαμβάνει ειδοποίηση ότι τα δεδομένα του αξιοποιήθηκαν σε κάποια συγκεκριμένη έρευνα ή μελέτη.

Συμπεράσματα

Κατά τη διάρκεια της διαδικασίας εκπόνησης της παρούσας πτυχιακής εργασίας, προέκυψαν διάφορα συμπεράσματα τα οποία παραθέτω στη συνέχεια:

Από την εξέταση της βιβλιογραφίας προκύπτει ότι ο διαβήτης έχει εξελιχθεί σε μια παγκόσμια πανδημία και διευρύνεται ανησυχητικά στον ανθρώπινο πληθυσμό του πλανήτη, λαμβάνοντας μεγάλες διαστάσεις, και απειλώντας όλο και περισσότερο την υγεία πολλών συνανθρώπων μας.

Οι συνέπειες αποτυχίας ελέγχου του διαβήτη ενδέχεται να είναι σοβαρές, ακόμα και απειλητικές για τη ζωή του ασθενούς.

Όσο και αν η επιστήμη έχει επικεντρώσει και συμβάλει στην πρόληψη και στον περιορισμό της, υπάρχει ακόμη πεδίο που μπορεί να αξιοποιηθεί προς αυτήν την κατεύθυνση, τον περιορισμό της νόσου.

Η εφαρμογή «Diabetic Data Solutions», για τους διαβητικούς, που προτείνω μέσω της εργασίας μου, και προορίζεται για χρήση από επαγγελματίες υγείας, θεωρώ ότι μπορεί να συμβάλει προς αυτήν την κατεύθυνση.

Σύμφωνα με την έρευνα που διεξήγαγα, δεν υπάρχει αυτή τη στιγμή κάποια καθιερωμένη πλατφόρμα ευρείας χρήσης, καταγραφής και παρακολούθησης των βιοδεικτών των πασχόντων από διαβήτη, η οποία να είναι προσβάσιμη στο ιατρικό δυναμικό, αλλά και τους ασθενείς.

Με την εφαρμογή, που αφορά ένα σύστημα διαχείρισης μητρώων ασθενών, πέραν της βοήθειας προς τους διαβητικούς και τους έχοντες την προδιάθεση της νόσου, τα συλλεγόμενα από την εφαρμογή στοιχεία δύναται να χρησιμοποιηθούν και για ερευνητικούς σκοπούς, συνεισφέροντας στην περαιτέρω μελέτη και αντιμετώπιση της ασθένειας.

Θεωρούμε αναγκαία την ύπαρξη ενός ευρέως αποδεκτού μοντέλου απαραίτητων βιοδεικτών, έτσι ώστε τα δεδομένα να μπορούν να χρησιμοποιηθούν με πιο αποτελεσματικό τρόπο κατά τη διάρκεια έρευνας πάνω στην ασθένεια.

Programming Journey

Ξεκινώντας την ανάπτυξη της εφαρμογής, το πρώτο βήμα ήταν να δημιουργήσω μια κενή εφαρμογή μέσω Vite, επιλέγοντας τις επιλογές React, JS και CSS. Μετά την επιτυχή δημιουργία, ως πρώτο βήμα καταγράφεται η δημιουργία του συστατικού που ουσιαστικά “περιγράφει” την οθόνη εγγραφής και εισόδου. Στη συνέχεια, μετά την προσθήκη λειτουργικότητας πλοήγησης μεταξύ σελίδων (routing), ως απαραίτητο βήμα καταγράφεται η παραμετροποίηση του firebase, έτσι ώστε να συνδεθεί με την εφαρμογή και να μας παρέχει την υπηρεσία αυθεντικοποίησης. Μετά την προσθήκη χειριστών συμβάντων (event handlers), ανάλογα με την επιτυχία ή αποτυχία της αυθεντικοποίησης, μπορούμε να χειριστούμε το γεγονός (event handling), και να προχωρήσουμε στη δημιουργία σελίδας υποδοχής, στην οποία μεταφέρεται ο χρήστης μετά από επιτυχημένη αυθεντικοποίηση. Συνεχίζοντας προσθέσαμε δύο διαδραστικά σχήματα στη σελίδα, πάνω στα οποία προσθέσαμε χειριστές συμβάντων, έτσι ώστε να μπορέσουμε να αντιδράσουμε στην επιλογή του χρήστη. Στη συνέχεια δημιουργήσαμε δύο καινούργια συστατικά τα οποία αντιπροσωπεύουν τα προαναφερθέντα κομμάτια της εφαρμογής, Drive και Registry, στα οποία πλοηγούνται μέσω των σχημάτων της προηγούμενης σελίδας. Για την πλοήγηση αξιοποιείται η λειτουργικότητα που παρέχεται από τη React, το react-router-dom. Σε αυτό το σημείο ξεκινά η ανάπτυξη των δύο μεγαλύτερων συστατικών, τα οποία περιέχουν όλες τις λειτουργικότητες της εφαρμογής μας.

Ως πρώτο βήμα για την ανάπτυξη του Drive, ορίσαμε μέσω html και css την δομή και εμφάνιση των placeholders για τα αρχεία, προσθέσαμε χειριστές συμβάντων, κουμπί μεταφόρτωσης αρχείων και δημιουργίας φακέλων και τα συνδέσαμε με το firebase. Εν συνεχεία προσθέσαμε τη λειτουργικότητα αναζήτησης αρχείων και μπάρας προόδου μαζί με κουμπί αποσύνδεσης από τη πλατφόρμα. Τέλος, προσθέσαμε κώδικα που αναγνωρίζει τα αρχεία από το file extension τους, και τους αποδίδει το κατάλληλο εικονίδιο κατά την εμφάνιση τους στην οθόνη.

Προχωρώντας με την ανάπτυξη του Registry, ορίσαμε την δομή του πίνακα ο οποίος θα περιέχει τα μελλοντικά στοιχεία των ασθενών μέσω HTML. Στη συνέχεια δημιουργήσαμε το αναδυόμενο παράθυρο μέσα στο οποίο προσθέσαμε την φόρμα με τα απαραίτητα πεδία προς καταχώρηση, και συνδέσαμε την επιτυχή συμπλήρωση και υποβολή (submit) της με την καταχώρηση και προβολή στοιχείων στο πίνακα. Επίσης, προσθέσαμε τη λειτουργία αναζήτησης καταχωρήσεων βάσει το όνομα, τη λειτουργία επεξεργασίας καταχωρήσεων με χρήση διπλού κλικ, ένα κουμπί για πλοήγηση στη προηγούμενη σελίδα και ένα κουμπί για αποσύνδεση από την εφαρμογή. Στη συνέχεια έγινε προσθήκη της σελίδας διαγραμμάτων,

όπου αφού επιλέξαμε τις κατάλληλες βιβλιοθήκες, μεταφέραμε τα δεδομένα από το Registry στη σελίδα Statistics και τέλος υλοποιήσαμε τα κουμπιά που αφορούν τις λειτουργικότητες φιλτραρίσματος των διαγραμμάτων και την πλοήγηση. Τέλος προσθέσαμε το κουμπί και τη λειτουργικότητα για λήψη του συνόλου των καταχωρήσεων σε μορφή csv, έχοντας πρώτα εξαλείψει όλα τα δεδομένα που μπορούν να θεωρηθούν αναγνωριστικά στοιχεία του ασθενούς.

References

1. Sapra A, Bhandari P. Diabetes. [Updated 2023 Jun 21]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2023 Jan-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK551501/>
2. Thota S, Akbar A. Insulin. [Updated 2023 Jul 10]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2023 Jan-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK560688/>
3. <https://www.who.int/news-room/fact-sheets/detail/diabetes>
4. Kharroubi AT, Darwish HM. Diabetes mellitus: The epidemic of the century. World J Diabetes. 2015 Jun 25;6(6):850-67. doi: 10.4239/wjd.v6.i6.850. PMID: 26131326; PMCID: PMC4478580.
5. Nazar CM, Bojerenu MM, Safdar M, Marwat J. Effectiveness of diabetes education and awareness of diabetes mellitus in combating diabetes in the United Kingdom; a literature review. J Nephropharmacol. 2015 Sep 9;5(2):110-115. PMID: 28197516; PMCID: PMC5297564.
6. Dorcelly B, Katz K, Jagannathan R, Chiang SS, Oluwadare B, Goldberg IJ, Bergman M. Novel biomarkers for prediabetes, diabetes, and associated complications. Diabetes Metab Syndr Obes. 2017 Aug 14;10:345-361. doi: 10.2147/DMSO.S100074. PMID: 28860833; PMCID: PMC5565252.
7. <https://www.diabetesresearchclinicalpractice.com/>
8. Watzlaf VJ, Zeng X, Jarymowycz C, Firouzan PA. Standards for the content of the electronic health record. Perspect Health Inf Manag. 2004 Jan 7;1:1. PMID: 18066381; PMCID: PMC2047330.
9. <https://gdpr-info.eu/issues/personal-data/>
10. Toh MP, Leong HS, Lim BK. Development of a diabetes registry to improve quality of care in the National Healthcare Group in Singapore. Ann Acad Med Singap. 2009 Jun;38(6):546-6. PMID: 19565107.
11. Rollo ME, Aguiar EJ, Williams RL, Wynne K, Kriss M, Callister R, Collins CE. eHealth technologies to support nutrition and physical activity behaviors in diabetes self-management. Diabetes Metab Syndr Obes. 2016 Nov 4;9:381-390. doi: 10.2147/DMSO.S95247. PMID: 27853384; PMCID: PMC5104301.
12. <https://www.overlaphealth.com/>
13. <https://www.vivifyhealth.com/>
14. <https://validic.com/>
15. <https://www.javascript.com/>
16. <https://react.dev/>
17. <https://html.com/>
18. <https://www.w3.org/Style/CSS/Overview.en.html>
19. <https://nodejs.org>
20. <https://firebase.google.com/>
21. <https://vitejs.dev/>
22. <https://www.npmjs.com/>
23. <https://threejs.org/>
24. <https://www.framer.com/motion/>
25. <https://tailwindcss.com/>
26. <https://getbootstrap.com>
27. <https://www.chartjs.org/>

APPENDIX

Auth.jsx (Αυθεντικοποίηση)

```
import React, { useEffect, useState } from 'react';
import { getAuth, signInWithEmailAndPassword, createUserWithEmailAndPassword,
onAuthStateChanged } from 'firebase/auth';
import { useNavigate } from 'react-router-dom';

import styles from './_Auth.module.scss';

import AuthWrapper from './AuthWrapper';

import WavySphereComponent from '../WavySphereComponent';

export default function Auth({ email, setEmail, password, setPassword, isRegistering,
setIsRegistering }) {
  let auth = getAuth();
  let navigate = useNavigate();

  const [errorMessage, setErrorMessage] = useState("");

  const handleEmailChange = (event) => setEmail(event.target.value);
  const handlePasswordChange = (event) => setPassword(event.target.value);

  const login = (event) => {
    event.preventDefault();
    signInWithEmailAndPassword(auth, email, password)
      .then((res) => {
        //console.log(res.user)
      })
      .catch((err) => {
        setErrorMessage(err.message);
      });
  };

  const register = (event) => {
    event.preventDefault();
    createUserWithEmailAndPassword(auth, email, password)
      .then((res) => {
        //console.log(res.user)
      })
  };
}
```

```

        .catch((err) => {
            setErrorMessage(err.message);
        });
    };

    useEffect(() => {
        onAuthStateChanged(auth, (user) => {
            //console.log(user)

            if (user) {
                navigate("/intro");
            } else {
                navigate("/");
            }
        });
    }, [auth, navigate]);

    return (
        <
            <div className={styles.containerauth}>
                <WavySphereComponent />
                <div className={styles.authbtn__container}>
                    <div className={styles.authbtn}>
                        
                        <h1 className={styles.authbtn__title}>Diabetic Data Solutions</h1>
                    </div>
                    <div className={styles.login__container}>
                        <form onSubmit={isRegistering ? register : login} className={styles.loginForm}>
                            <input
                                type="email"
                                placeholder="Email"
                                value={email}
                                onChange={handleEmailChange}
                                className={styles.loginInput}
                                required
                            />
                            <input
                                type="password"
                                placeholder="Password"
                                value={password}
                                onChange={handlePasswordChange}
                                className={styles.loginInput}
                                required
                            />
                            <button type="submit" className={styles.authbtn__loginbutton}>

```

```

        {isRegistering ? "Register" : "Login"}
      </button>
    </form>
    {errorMessage && <p className={styles.errorMsg}>{errorMessage}</p>}
    <button
      onClick={() => setIsRegistering(!isRegistering)}
      className={styles.authbtn__togglebutton}
    >
      {isRegistering ? "Already have an account? Login" : "Don't have an account?
Register"}
    </button>
  </div>
  { /* <Footer /> */ }
</div>
</div>
</div>
</>
);
}

```

AnimatedRoutes.jsx (Αρχείο πλοήγησης)

```

import React, {Suspense} from "react";
import { Routes, Route, useLocation } from "react-router-dom";

import { AnimatePresence } from "framer-motion";

import Drive from './Drive';
import Auth from './Auth';
import Folder from './Folder';
import Intro from './Intro';
import AuthWrapper from './AuthWrapper';
import Stats from './Stats';

import {database} from './firebaseConfig'

import Register from './Register';

function AnimatedRoutes() {
  const location = useLocation();

  return (

    <AnimatePresence>

```

```

<Suspense fallback={null}/>
<Routes key={location.pathname} location={location}>

  <Route path="/" element={<AuthWrapper database={database} />} />
  <Route path="/drive" element={<Drive database={database} />} />
  <Route path="/intro" element={<Intro database={database} />} />
  <Route path="/register" element={<Register database={database} />} />
  <Route path="/folder/:id" element={<Folder database={database} />} />
  <Route path="/stats" element={<Stats database={database} />} />
</Routes>
</AnimatePresence>
);
}

export default AnimatedRoutes;

```

Registry.jsx (Μητρώο Ασθενών)

```

import React, { useState, useEffect, useRef } from "react";
import Modal from "../Modal";
import { FiArrowLeftCircle } from "react-icons/fi";
import { useNavigate } from "react-router-dom";
import EditableCell from "../EditableCell";

import { AiFillEye, AiFillEyeInvisible } from "react-icons/ai";
import { signInOut, getAuth, onAuthStateChanged } from "firebase/auth";

import { motion } from "framer-motion";
import { Col, Row } from "react-bootstrap";

export default function Register() {
  const [data, setData] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [showModal, setShowModal] = useState(false);
  const [editIndex, setEditIndex] = useState(null);
  const [isSubmitting, setIsSubmitting] = useState(false);
  const enterPressed = useRef(false);
  const [isEditing, setIsEditing] = useState(false);
  const [updatedValue, setUpdatedValue] = useState("");
  const [isVisible, setIsVisible] = useState(false);

```

```
let auth = getAuth();
```

```
const user = auth.currentUser;  
const navigate = useNavigate();
```

```
const useAuthState = () => {  
  const navigate = useNavigate();
```

```
  useEffect(() => {  
    const unsubscribe = onAuthStateChanged(auth, (user) => {  
      if (user) {  
        navigate("/register");  
      } else {  
        navigate("/");  
      }  
    });  
  });
```

```
  return () => {  
    unsubscribe();  
  };  
}, [navigate]);  
};
```

```
useAuthState();
```

```
useEffect(() => {  
  const storedData = localStorage.getItem("rowData");  
  
  if (storedData) {  
    setData(JSON.parse(storedData));  
  }  
}, []);
```

```
const saveData = (ev) => {  
  if (ev) {  
    ev.preventDefault();
```

```

}
const form = document.getElementById("myForm");
const formData = new FormData(form);
const date = formData.get("date");
const inputDate = formData.get("date");
const formattedDate = formatDate(date); // Format the date

const firstLastName = formData.get("flname");
const age = formData.get("age");
const weight = formData.get("weight");
const height = formData.get("height");
const gender = formData.get("gender");

const glucoseAfterFasting = formData.get("glaf");
const hba1c = formData.get("hba1c");
const glucoseMouth1 = formData.get("glm1");
const glucoseMouth2 = formData.get("glm2");

const newRow = [
  firstLastName,
  age,
  weight,
  height,
  gender,
  formattedDate,
  glucoseAfterFasting,
  hba1c,
  glucoseMouth1,
  glucoseMouth2,
];

setData((prevData) => {
  const newData = [...prevData, newRow];
  const sortedData = newData.sort((a, b) => {
    const dateA = new Date(a[5]);
    const dateB = new Date(b[5]);
    if (dateA > dateB) {
      return -1;
    } else if (dateA < dateB) {
      return 1;
    } else {
      return 0;
    }
  });
});

```

```

// Handle the case where two rows have the same first and last name
const sameNameRows = sortedData.filter((row) => row[0] === firstLastName);
if (sameNameRows.length > 1) {
  const sameNameSortedData = sameNameRows.sort((a, b) => {
    const dateA = new Date(a[5]);
    const dateB = new Date(b[5]);
    if (dateA > dateB) {
      return -1;
    } else if (dateA < dateB) {
      return 1;
    } else {
      return 0;
    }
  });

  // Replace the unsorted rows with the sorted rows
  const index = sortedData.indexOf(sameNameRows[0]);
  const endIndex = index + sameNameRows.length;
  sortedData.splice(index, endIndex - index, ...sameNameSortedData);
}

localStorage.setItem("rowData", JSON.stringify(sortedData));
return sortedData;
});

closeModal();
};

const handleEnter = useEnterKey(() =>
  handleEditKeyDown(editedRow, editedCol)
);

function useEnterKey(callback) {
  const handleKeyDown = (event) => {
    if (event.key === "Enter") {
      event.preventDefault();
      callback();
    }
  };
};

return handleKeyDown;
}

```

```

const editCell = (event) => {
  const rowIndex = event.target.parentNode.rowIndex - 1;
  const colIndex = event.target.cellIndex;
  setEditIndex([rowIndex, colIndex]);
  setIsEditing(true);
};

const handleEditKeyDown = (event, rowIndex, colIndex, value) => {
  if (event.key === "Enter") {
    event.preventDefault();
    const newData = [...data];
    newData[rowIndex][colIndex] = value;
    setData(newData);
    localStorage.setItem("rowData", JSON.stringify(newData));
    setEditIndex(null);
    setIsEditing(false);
  }
};

const [editedRow, setEditedRow] = useState(null);

const goBack = () => {
  navigate("/intro");
};

const cacheData = (formElement) => {
  if (!formElement || !(formElement instanceof HTMLFormElement)) {
    return;
  }
  const formData = new FormData(formElement);
  const date = formData.get("date");
  const inputDate = formData.get("date");
  const formattedDate = date ? formatDate(date) : "";
  // check if date is not empty before formatting it

  const firstLastName = formData.get("flname");
  const age = formData.get("age");
  const weight = formData.get("weight");
  const height = formData.get("height");
  const gender = formData.get("gender");

  const glucoseAfterFasting = formData.get("glaf");
  const hba1c = formData.get("hba1c");
  const glucoseMouth1 = formData.get("glm1");

```

```

const glucoseMouth2 = formData.get("glm2");

const newRow = [
  firstLastName,
  age,
  weight,
  height,
  gender,
  formattedDate, // set formattedDate to an empty string if date is empty
  glucoseAfterFasting,
  hba1c,
  glucoseMouth1,
  glucoseMouth2,
];

setData((prevData) => {
  const newData = [...prevData, newRow];
  localStorage.setItem("rowData", JSON.stringify(newData));
  return newData;
});

closeModal();
};

const headers = [
  "fullName",
  "age",
  "weight",
  "height",
  "gender",
  "date",
  "glucoseAfterFasting",
  "hba1c",
  "glucoseMouth1",
  "glucoseMouth2",
];

const downloadCSV = () => {
  const headers = [
    "age",
    "weight",
    "height",
    "gender",
    "date",

```

```

    "glucoseAfterFasting",
    "hba1c",
    "glucoseMouth1",
    "glucoseMouth2",
  ];

  const filteredData = data.map((row) => {
    const [, ...restOfRow] = row;
    return restOfRow;
  });

  const csvContent =
    "data:text/csv;charset=utf-8," +
    [headers.join(","), ...filteredData.map((row) => row.join(","))].join("\n");

  const encodedUri = encodeURI(csvContent);
  const link = document.createElement("a");
  link.setAttribute("href", encodedUri);
  link.setAttribute("download", "data.csv");
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
};

const colordata = [
  {
    red: "#e2625e",
    blue: "#7f9ae4",
    green: "#7fe47f",
    yellow: "#e2d95e",
    orange: "#e68f56"
  }
]

const toggle = () => {
  setIsVisible(!isVisible);
}

const openModal = () => {
  setShowModal(true);
}

```

```

};

const closeModal = () => {
  setShowModal(false);
};

const gotostats = () => {
  navigate("/stats");
}

const formatDate = (dateString) => {
  const date = new Date(dateString);
  const day = String(date.getDate()).padStart(2, "0");
  const month = String(date.getMonth() + 1).padStart(2, "0");
  const year = date.getFullYear();

  return `${day}/${month}/${year}`;
};

return (
  <div className="top-bar">
    <div className="back-icon">
      <FiArrowLeftCircle size={50} onClick={goBack} />
    </div>
    <div className="registerbuttonscontainer">
      <button className="entry-button" onClick={openModal}>
        <img alt="plus icon" data-bbox="165 605 185 620"/> Add New Entry
      </button>

      <div id="display">
        <div className="searchdiv">
          <input

            type="text"
            placeholder="Search..."
            onChange={(e) => setSearchTerm(e.target.value)}
          />

          <button className="hidecells" onClick={toggle}>
            {isVisible ? <AiFillEyeInvisible size={35} style={{ color: "black" }} /> : <AiFillEye
            size={35} style={{ color: "black" }} />}
          </button>
        </div>
      </div>
    </div>
  )

```

```
</button>
```

```
</div>
```

```
</div>
```

```
<button className="download-button" onClick={downloadCSV}>
```

```
 Download CSV
```

```
</button>
```

```
<button className="download-button" onClick={gotostats}>
```

```
 Charts
```

```
</button>
```

```
</div>
```

```
<Modal showModal={showModal} closeModal={closeModal}>
```

```
<div id="collect">
```

```
<form id="myForm" onSubmit={saveData}>
```

```
<div className="con1">
```

```
<div className="input-container">
```

```
<label className="inp inputvalidation1" htmlFor="1">
```

```
<input
```

```
placeholder="&nbsp;"
```

```
className=""
```

```
type="text"
```

```
id="fname"
```

```
name="fname"
```

```
required
```

```
<span className="label">First/Last Name</span>
```

```
<span className="focus-bg"></span>
```

```
</label>
```

```
</div>
```

```
<div className="input-container">
```

```
<label className="inp inputvalidation2" htmlFor="2">
```

```
<input
```

```
placeholder="&nbsp;"
```

```
className="clean-slide"
```

```
type="text"
```

```
id="age"
```

```

        name="age"
        required
        pattern="[0-9]*"
    />
    <span className="label">Age</span>
    <span className="focus-bg"></span>
</label>
</div>
<div className="input-container">
    <label className="inp inputvalidation3" htmlFor="3">
    <input
        placeholder="&nbsp;"
        className="clean-slide"
        type="text"
        id="weight"
        name="weight"
        required
        pattern="[0-9]*"
    />
    <span className="label">Weight</span>
    <span className="focus-bg"></span>
</label>
</div>
<div className="input-container">
    <label className="inp inputvalidation4" htmlFor="4">
    <input
        placeholder="&nbsp;"
        className="clean-slide"
        type="text"
        id="height"
        name="height"
        required
        pattern="[0-9]*"
    />
    <span className="label">Height</span>
    <span className="focus-bg"></span>
</label>
</div>
</div>

```

```

<div className="con2">
  <div className="input-container">
    <label htmlFor="5">Gender</label>
    <select className="selectgender" id="gender" name="gender">
      <option value="Male">Male</option>
      <option value="Female">Female</option>
    </select>
  </div>
  <div className="input-container">
    <label htmlFor="5">Date</label>
    <input className="selectdate" type="date" id="date" name="date" required />
  </div>
</div>

```

```

<div className="con3">
  <div className="input-container">
    <label className="inp inputvalidation5" htmlFor="7">
      <input
        placeholder="&nbsp;"
        type="number"
        id="glaf"
        name="glaf"
        min="0"
        max="300"
        step="1"
        required
      />
      <span className="label">gAF</span>
      <span className="focus-bg"></span>
    </label>
  </div>
  <div className="input-container">
    <label className="inp inputvalidation6" htmlFor="8">
      <input
        placeholder="&nbsp;"
        type="number"
        id="hba1c"
        name="hba1c"
        min="0"

```

```

        max="20"
        step="0.1"
        required
    />
    <span className="label">hba1C</span>
    <span className="focus-bg"></span>
  </label>
</div>
<div className="input-container">
  <label className="inp inputvalidation7" htmlFor="9">
    <input
      placeholder="&nbsp;"
      type="number"
      id="glm1"
      name="glm1"
      min="0"
      max="200"
      step="1"
    />
    <span className="label">gM1</span>
    <span className="focus-bg"></span>
  </label>
</div>
<div className="input-container">
  <label className="inp inputvalidation8" htmlFor="10">
    <input
      placeholder="&nbsp;"
      type="number"
      id="gml2"
      name="glm2"
      min="0"
      max="200"
      step="1"
    />
    <span className="label">gM2</span>
    <span className="focus-bg"></span>
  </label>
</div>
</div>
<div className="con4">
  <button id="btnSave" type="submit">
    Save 📁
  </button>
</div>

```

```

</form>
</div>
</Modal>

```

```

{isVisible && (
<table >
  <thead>
    <tr>
      {headers.map((header, colIndex) => (
        <th key={colIndex}>{header}</th>
      ))}
    </tr>
  </thead>
  <tbody>
    {data
      .filter((row) => {
        const values = row.join("").toLowerCase();
        return values.includes(searchTerm.toLowerCase());
      })
      .map((row, rowIndex) => (
        <tr key={rowIndex}>
          {row.map((cell, colIndex) => (
            <td
              key={colIndex}
              onDoubleClick={(event) => {
                editCell(event);
              }}
              style={{
                backgroundColor:
                  colIndex === 6 &&
                  !(
                    editIndex !== null &&
                    editIndex[0] === rowIndex &&
                    editIndex[1] === colIndex
                  )
                ? parseFloat(cell) >= 0 && parseFloat(cell) <= 80
                  ? "#f9ae4"
                  : parseFloat(cell) >= 81 &&
                    parseFloat(cell) <= 115
                  ? "#fe47f"
                  : parseFloat(cell) > 115
                  ? "#e2625e"

```

```

: ""
: colIndex === 8 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell) >= 0 && parseFloat(cell) <= 110
? "#7f9ae4"
: parseFloat(cell) >= 111 &&
  parseFloat(cell) <= 170
? "#7fe47f"
: parseFloat(cell) > 171
? "#e2625e"
: ""

```

```

: colIndex === 5 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell)
? "#f3e2df"

```

```

: ""

```

```

: colIndex === 4 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? "#f3e2df"

```

```

: colIndex === 3 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)

```

```

? parseFloat(cell)
? "#f3e2df"

: ""

: colIndex === 2 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell)
? "#f3e2df"

: ""

: colIndex === 1 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell)
? "#f3e2df"

: ""

: colIndex === 0 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell)
? "#f3e2df"

: ""

: colIndex === 9 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)

```

```

? parseFloat(cell) >= 0 && parseFloat(cell) <= 69
? "#7f9ae4"
: parseFloat(cell) >= 70 &&
  parseFloat(cell) <= 120
? "#7fe47f"
: parseFloat(cell) > 120
? "#e2625e"
: ""
: colIndex === 7 &&
!(
  editIndex !== null &&
  editIndex[0] === rowIndex &&
  editIndex[1] === colIndex
)
? parseFloat(cell) >= 0 && parseFloat(cell) <= 6
? "#7fe47f"
: parseFloat(cell) > 6.1 &&
  parseFloat(cell) <= 6.5
? "#e2d95e"
: parseFloat(cell) > 6.6 &&
  parseFloat(cell) <= 8.5
? "#e68f56"
: parseFloat(cell) > 8.6
? "#e2625e"
: ""
: "",
}}
>
{editIndex !== null &&
editIndex[0] === rowIndex &&
editIndex[1] === colIndex ? (
<EditableCell
  key={`${rowIndex}-${colIndex}`}
  cell={cell}
  editedRow={editIndex !== null ? editIndex[0] : -1}
  editedCol={editIndex !== null ? editIndex[1] : -1}
  handleEditKeyDown={handleEditKeyDown}
/>
): (
  cell
)}
</td>
))}
</tr>

```

```

    ))}
  </tbody>
</table>
)}
</div>

</>
);
}

```

Stats.jsx (Διαγράμματα)

```

import React, { useState } from 'react';
import { Pie } from 'react-chartjs-2';
import { Chart, ArcElement, CategoryScale, DoughnutController } from 'chart.js';
import ChartDataLabels from 'chartjs-plugin-datalabels';
import { FiArrowLeftCircle, FiLogout } from "react-icons/fi";
import { useNavigate } from "react-router-dom";
import { signIn, getAuth, onAuthStateChanged } from "firebase/auth";

// Register the necessary components and plugins with Chart.js
Chart.register(ArcElement, CategoryScale, DoughnutController, ChartDataLabels);

const headers = [
  "age", "weight", "height", "gender", "date",
  "glucoseAfterFasting", "hba1c", "glucoseMouth1", "glucoseMouth2"
];

export default function Stats() {
  const storedData = JSON.parse(localStorage.getItem("rowData"));
  let auth = getAuth();
  const user = auth.currentUser;
  const navigate = useNavigate();
  const [flag, setFlag] = useState(false);
  const [filterLessThan, setFilterLessThan] = useState(false);

  const goBack = () => {
    navigate("/register");
  };

  const handleSignOut = () => {
    signIn(auth).then(() => {
      navigate("/login");
    }).catch((error) => {

```

```

        console.error("Error signing out: ", error);
    });
};

const [thresholds, setThresholds] = useState({
    age: "",
    weight: "",
    height: "",
    glucoseAfterFasting: "",
    hba1c: "",
    glucoseMouth1: "",
    glucoseMouth2: ""
});

const [showModal, setShowModal] = useState(false);

const handleSubmit = (e) => {
    e.preventDefault();
    setShowModal(false);
};

const generateChartData = (headerIndex) => {
    let dataForHeader = Object.values(storedData).map(entry => entry[headerIndex + 1]);

    if (thresholds[headers[headerIndex]] && headers[headerIndex] !== "gender" &&
headers[headerIndex] !== "date") {
        const threshold = thresholds[headers[headerIndex]];
        if(!flag) { // Corrected the if condition
            dataForHeader = dataForHeader.filter(value => parseFloat(value) >=
parseFloat(threshold));
        } else {
            dataForHeader = dataForHeader.filter(value => parseFloat(value) <=
parseFloat(threshold));
        }
    }
}

if (dataForHeader.length === 0) {
    return {
        labels: ["No Data"],
        datasets: [{
            data: [1], // placeholder value
            backgroundColor: ['#E7E9ED'],

```

```

        hoverBackgroundColor: ['#E7E9ED']
      }]
    };
  }

  let counts = {};
  dataForHeader.forEach(item => {
    counts[item] = (counts[item] || 0) + 1;
  });

  let colors = ['#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0', '#E7E9ED'];

  if (headers[headerIndex] === 'gender') {
    colors = Object.keys(counts).map(key => {
      if (key.toLowerCase() === 'female') return '#FF6384';
      if (key.toLowerCase() === 'male') return '#36A2EB';
      return '#E7E9ED';
    });
  }

  return {
    labels: Object.keys(counts),
    datasets: [{
      data: Object.values(counts),
      backgroundColor: colors,
      hoverBackgroundColor: colors
    }]
  };
};

const chartOptions = {
  plugins: {
    legend: {
      display: true,
      position: 'top'
    },
    datalabels: {
      display: true,
      formatter: (value, ctx) => {
        const label = ctx.chart.data.labels[ctx.dataIndex];
        if (ctx.datasetIndex === 0) {
          return label;
        }
      }
    }
  }
};

```

```

    } else {
      return value > 1 ? value : "";
    }
  },
  color: 'white',
  anchor: 'center',
  align: (context) => {
    return context.datasetIndex === 0 ? 'center' : 'end';
  },
  offset: (context) => {
    return context.datasetIndex === 0 ? 0 : 20;
  }
},
tooltip: {
  callbacks: {
    label: function(context) {
      const label = context.label || "";
      const value = context.parsed;
      return `${label}: ${value}`;
    }
  }
}
};

```

```

const resetFilters = () => {
  setThresholds({
    age: "",
    weight: "",
    height: "",
    glucoseAfterFasting: "",
    hba1c: "",
    glucoseMouth1: "",
    glucoseMouth2: ""
  });
};

```

```

const togglethresholds= () => {
  setFlag(prevFlag => !prevFlag); // Toggle the flag state
}

```

```

return (
  <div>
    <div className="back-icon">

```

```

<FiArrowLeftCircle size={50} onClick={goBack} />
</div>
<div style={{ display: 'flex', flexWrap: 'wrap', justifyContent: 'space-evenly' }}>
  {headers.map((header, index) => (
    <div key={header} style={{ minWidth: '200px', maxWidth: '300px', margin:
'1em', flexBasis: 'calc(33.333% - 2em)' }}>
      <h2 style={{ textAlign: 'center' }}>{header} Entries</h2>
      <Pie data={generateChartData(index)} options={chartOptions} />
    </div>
  ))}
</div>
<button style={{
  background: 'linear-gradient(to right, #FF6384, #36A2EB)',
  border: 'none',
  color: 'white',
  padding: '10px 20px',
  borderRadius: '5px',
  fontWeight: 'bold',
  cursor: 'pointer',
  outline: 'none',
  transition: 'background 0.3s, transform 0.3s'
}} onClick={() => setShowModal(true)} onMouseOver={e =>
e.currentTarget.style.background = 'linear-gradient(to right, #36A2EB, #FF6384)'}
onMouseOut={e => e.currentTarget.style.background = 'linear-gradient(to right,
#FF6384, #36A2EB)'}>
  Apply filters</button>

<button
  style={{
    background: 'linear-gradient(to right, #FF6384, #36A2EB)',
    border: 'none',
    color: 'white',
    padding: '10px 20px',
    borderRadius: '5px',
    fontWeight: 'bold',
    cursor: 'pointer',
    outline: 'none',
    transition: 'background 0.3s, transform 0.3s',
    flex: '1',
    marginRight: '10px'
  }}
  onClick={resetFilters}
  onMouseOver={e => e.currentTarget.style.background = 'linear-gradient(to right,
#36A2EB, #FF6384)'}

```

```

    onMouseOut={e => e.currentTarget.style.background = 'linear-gradient(to right, #FF6384,
#36A2EB)'}
  >
    Reset filters
  </button>

  {showModal && (
    <div style={{ position: 'fixed', top: '0', left: '0', right: '0', bottom: '0', background:
'rgba(0, 0, 0, 0.5)', zIndex: '999' }} onClick={() => setShowModal(false)}></div>
    <div style={{
      position: 'fixed', top: '50%', left: '50%', transform: 'translate(-50%, -50%)',
      backgroundColor: 'white', padding: '20px', boxShadow: '0px 0px 10px
rgba(0,0,0,0.1)', zIndex: '1000', borderRadius: '10px'
    }}>
      <form onSubmit={handleSubmit}>
        {Object.keys(thresholds).map(key => (
          <div key={key} style={{ marginBottom: '15px' }}>
            <label style={{ display: 'block', marginBottom: '5px' }}>
              {key.charAt(0).toUpperCase() + key.slice(1)} Threshold:
            </label>
            <input
              type="number"
              style={{ padding: '10px', border: '1px solid #ddd', borderRadius:
'5px', width: '100%' }}
              value={thresholds[key]}
              onChange={e => setThresholds(prev => ({ ...prev, [key]:
e.target.value })))}
            />
          </div>
        ))}
      <div style={{ display: 'flex', justifyContent: 'space-between' }}>
        <button type="submit" style={{
          background: 'linear-gradient(to right, #FF6384, #36A2EB)',
          border: 'none',
          color: 'white',
          padding: '10px 20px',
          borderRadius: '5px',
          fontWeight: 'bold',
          cursor: 'pointer',
          outline: 'none',
          transition: 'background 0.3s, transform 0.3s',
          flex: '1',
          marginRight: '10px'

```

```

    }} onMouseOver={e => e.currentTarget.style.background = 'linear-
gradient(to right, #36A2EB, #FF6384)'} onMouseOut={e =>
e.currentTarget.style.background = 'linear-gradient(to right, #FF6384,
#36A2EB)'}>Apply</button>

```

```

<button type= 'button' style={{
  background: 'linear-gradient(to right, #FF6384, #36A2EB)',
  border: 'none',
  color: 'white',
  padding: '10px 20px',
  borderRadius: '5px',
  fontWeight: 'bold',
  cursor: 'pointer',
  outline: 'none',
  transition: 'background 0.3s, transform 0.3s',
  flex: '1',
  marginRight: '10px'
}} onMouseOver={e => e.currentTarget.style.background = 'linear-
gradient(to right, #36A2EB, #FF6384)'}
onMouseOut={e => e.currentTarget.style.background = 'linear-
gradient(to right, #FF6384, #36A2EB)'}
onClick={togglethresholds}
>{!flag ? ">=" : "<="}</button>
<button type="button" onClick={() => setShowModal(false)} style={{
  background: '#ddd',
  border: 'none',
  color: 'red',
  padding: '10px 20px',
  borderRadius: '5px',
  fontWeight: 'bold',
  cursor: 'pointer',
  outline: 'none',
  transition: 'background 0.3s, transform 0.3s',
  flex: '1',
  marginLeft: '10px'
}}
onMouseOver={e => e.currentTarget.style.background = '#bbb'}
onMouseOut={e => e.currentTarget.style.background = '#ddd'}
>Close</button>
</div>
</form>
</div>
</>

```

```

    })
  </div>
);
}

```

Drive.jsx (χώρος αποθήκευσης αρχείων)

```

import React, { useEffect, useState } from "react";
import { FcOpenedFolder } from "react-icons/fc";
import { FiFilePlus, FiLogOut, FiArrowLeftCircle } from "react-icons/fi";
import { Modal, Input } from "antd";
import {
  collection,
  addDoc,
  onSnapshot,
  doc,
  deleteDoc,
  getDoc,
  updateDoc,
} from "firebase/firestore";
import { useNavigate } from "react-router-dom";
import { signOut, getAuth, onAuthStateChanged } from "firebase/auth";
import {
  getStorage,
  ref,
  uploadBytes,
  uploadBytesResumable,
  getDownloadURL,
} from "firebase/storage";
import ProgressBar from "./ProgressBar";
import { motion } from "framer-motion";
import Container from "react-bootstrap/Container";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import { FaFileExcel, FaFileImage, FaFilePdf, FaFileWord, FaFile } from "react-icons/fa";
import { FaFolder } from "react-icons/fa";

export default function Drive({ database }) {
  let navigate = useNavigate();

```

```

let auth = getAuth();

const user = auth.currentUser;

const storage = getStorage();
const [progress, setProgress] = useState(null);
const collectionRef = collection(database, "driveData");
const [folderName, setFolderName] = useState("");
const [folders, setFolders] = useState([]);
const [isModalOpen, setIsModalOpen] = useState(false);
const [searchTerm, setSearchTerm] = useState("");
const [folderToDelete, setFolderToDelete] = useState(null);

const getFileIcon = (fileName) => {
  const fileExtension = fileName.split(".").pop().toLowerCase();

  switch (fileExtension) {
    case "jpg":
    case "jpeg":
    case "png":
    case "gif":
      return <FaFileImage className="drive__fileicon drive__fileicon__imgs" />;
    case "pdf":
      return <FaFilePdf className="drive__fileicon drive__fileicon__pdf" />;
    case "csv":
      return <FaFileExcel className="drive__fileicon drive__fileicon__excel" />;
    case "doc":
    case "docx":
      return <FaFileWord className="drive__fileicon drive__fileicon__word" />;
    default:
      return <FaFile className="drive__fileicon drive__fileicon__file" />;
  }
};

const [contextMenu, setContextMenu] = useState({
  visible: false,
  x: 0,
  y: 0,
  folderId: null,
});

const showContextMenu = (event, folderId) => {

```

```

event.preventDefault();

setContextMenu({
  visible: true,
  x: event.clientX,
  y: event.clientY,
  folderId,
});
};

const hideContextMenu = () => {
  setContextMenu({
    ...contextMenu,
    visible: false,
  });
};

const copy = async (src, dest) => {
  const srcUrl = await getDownloadURL(src);
  const response = await fetch(srcUrl);
  const blob = await response.blob();
  await uploadBytes(dest, blob);
};

const renameFolder = async (folderId, newName) => {
  try {
    const docRef = doc(database, "driveData", folderId);
    const folder = await getDoc(docRef);

    if (folder.data().folderName) {
      await updateDoc(docRef, { folderName: newName });
    } else if (folder.data().fileName) {
      const oldFileRef = ref(storage, folder.data().fileName);
      const newFileRef = ref(storage, newName);

      // Clone the file
      copy(oldFileRef, newFileRef)
        .then(async () => {
          // Get the new download URL
          const downloadURL = await getDownloadURL(newFileRef);

          // Update the database entry
          await updateDoc(docRef, {
            fileName: newName,

```

```

        downloadURL: downloadURL,
    });

    // Delete the old file from storage
    deleteObject(oldFileRef);
  })
  .catch((error) => {
    console.error("Error copying file in storage", error);
  });
}
} catch (error) {
  console.error("Error renaming folder/file", error);
}
};

const showModal = () => {
  setIsModalOpen(true);
};

const handleCancel = () => {
  setIsModalOpen(false);
};

const folderUpload = () => {
  addDoc(collectionRef, {
    folderName: folderName,
    fileLink: [
      {
        downloadURL: "",
        fileName: "",
      },
    ],
  })
  .then(() => {
    setIsModalOpen(false);
    setFolderName("");
  })
  .catch((err) => {
    alert(err.message);
  });
};

```

```

const getFile = (event) => {
  const fileRef = ref(storage, event.target.files[0].name);
  const uploadTask = uploadBytesResumable(fileRef, event.target.files[0]);

  uploadTask.on(
    "state_changed",
    (snapshot) => {
      const progress =
        (snapshot.bytesTransferred / snapshot.totalBytes) * 100;

      setProgress(Math.round(progress));
      // eslint-disable-next-line
      switch (snapshot.state) {
        case "paused":
          console.log("Upload is paused");
          break;
        case "running":
          console.log("Upload is running");
          break;
        // eslint-disable-next-line
      }
    },
    (error) => {},
    () => {
      getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {
        setProgress(null);
        addDoc(collectionRef, {
          fileName: event.target.files[0].name,
          downloadURL: downloadURL,
        });
      });
    }
  );
};

const readData = () => {
  onSnapshot(collectionRef, (data) => {
    setFolders(
      data.docs.map((doc) => {
        return { ...doc.data(), id: doc.id };
      })
    );
  });
};

```

```

const openFolder = (id) => {
  setFolderToDelete(null);
  navigate(`/folder/${id}`);
};

const openFile = (downloadURL) => {
  window.open(downloadURL, "_blank");
};

const logOut = () => {
  signOut(auth)
    .then(() => {
      navigate("/");
    })
    .catch((err) => {
      alert(err.message);
    });
};

const authState = () => {
  onAuthStateChanged(auth, (user) => {
    if (user) {
      navigate("/drive");
    } else {
      navigate("/");
    }
  });
};

useEffect(() => {
  readData();
  authState();

  if (folderToDelete) {
    if (window.confirm("Are you sure you want to delete this folder?")) {
      deleteFolder(folderToDelete);
    }
    setFolderToDelete(null);
  }
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, [folderToDelete]);

```

```
const deleteFolder = async (folderId) => {
  try {
    const docRef = doc(database, "driveData", folderId);
    await deleteDoc(docRef);
  } catch (error) {
    console.error("Error deleting folder", error);
  }
};
```

```
const filteredFolders =
  folders && folders.length > 0
  ? folders.filter(
    (folder) =>
      folder.folderName
        ?.toLowerCase()
        .includes(searchTerm?.toLowerCase() ?? "") ||
      folder.fileName
        ?.toLowerCase()
        .includes(searchTerm?.toLowerCase() ?? "")
  )
  : [];
```

```
const goBack = () => {
  navigate("/intro");
};
```

```
return (
  <>
```

```
<div className="title-container">
  <FiArrowLeftCircle className="drivebackbutton" size={50} onClick={goBack} />
  
```

```
<Col>
  <p> Welcome </p>
  <h1 className="username_title">{user !== null ? user.email.split('@')[0] :
  <></>}</h1>
</Col>
```

```
</div>
```

```

<div className="box"></div>
<div className="icon-container">
  <Input
    className="search-bar"
    placeholder="Search files"
    value={searchTerm}
    onChange={(event) => setSearchTerm(event.target.value)}
    style={{ zIndex: 100, width: 500 }}
  />

  <div className="upload-btn-wrapper">
    <FiFilePlus color="#18F2B2" className="icon" size={50} />
    <input type="file" className="myfile" onChange={getFile} />
  </div>
  <FcOpenedFolder size={50} className="icon" onClick={showModal} />
  <FiLogOut size={50} className="icon" color="#02020A" onClick={logout} />
</div>

<div className="progress-bar-container">
  {progress ? <ProgressBar completed={progress} /> : <</>}
</div>
<div className="grid-parent">
  {contextMenu.visible && (
    <div
      className="context-menu"
      style={{
        top: contextMenu.y,
        left: contextMenu.x,
        position: "fixed",
        zIndex: 1000,
      }}
      onClick={() => hideContextMenu()}
    >
      <ul>
        <li
          onClick={() => {
            const newName = prompt("Enter the new folder name");
            if (newName) {
              renameFolder(contextMenu.folderId, newName);
            }
            hideContextMenu();
          }}
        >

```

```

      Rename
    </li>
    <li>
      onClick={() => {
        if (
          window.confirm(
            "Are you sure you want to delete this folder?"
          )
        ) {
          deleteFolder(contextMenu.folderId);
        }
        hideContextMenu();
      }}
    >
      Delete
    </li>
  </ul>
</div>
))}

```

```

{filteredFolders.map((folder) => (
  <div
    className="grid-child"
    key={folder.id}
    onClick={() =>
      folder.folderName
        ? openFolder(folder.id)
        : openFile(folder.downloadURL)
    }
    onContextMenu={(event) => {
      showContextMenu(event, folder.id);
    }}
  >
    {folder.folderName ? <FaFolder className="drive__fileicon drive__fileicon__folder" /> :
    folder.fileName && getFileInfo(folder.fileName)}
    <h4>{folder.folderName ? folder.folderName : folder.fileName}</h4>
  </div>
))}

```

```

</div>
<Modal
  title="Folder Upload"
  open={isModalOpen}
  onOk={folderUpload}

```

```

      onCancel={handleCancel}
      centered
    >
      <Input
        placeholder="Enter the Folder Name"
        onChange={(event) => setFolderName(event.target.value)}
        value={folderName}
      />
    </Modal>
  </>
);
}

```

Folder.jsx (Αρχείο φακέλων)

```

import React, { useEffect, useState } from "react";
import { FiFilePlus, FiArrowLeftCircle } from "react-icons/fi";
import { useParams, useNavigate } from "react-router-dom";
import {
  getStorage,
  ref,
  uploadBytesResumable,
  getDownloadURL,
} from "firebase/storage";
import { updateDoc, doc, onSnapshot } from "firebase/firestore";
import ProgressBar from "../ProgressBar";
import { motion } from "framer-motion";

export default function Folder({ database }) {
  const storage = getStorage();
  let navigate = useNavigate();
  let params = useParams();
  const [folders, setFolders] = useState([]);
  const [progress, setProgress] = useState(null);
  const [folderName, setFolderName] = useState("");
  const databaseRef = doc(database, "driveData", params?.id);
  const [showContextMenu, setShowContextMenu] = useState(false);
  const [selectedFile, setSelectedFile] = useState({ file: null, position: { x: 0, y: 0 } });

  const openContextMenu = (event, file) => {
    event.preventDefault();
    setSelectedFile({

```

```

    file,
    position: { x: event.clientX, y: event.clientY },
  });
  setShowContextMenu(true);
};

const closeContextMenu = () => {
  setShowContextMenu(false);
};

const handleRename = () => {
  const newName = prompt("Enter new name:", selectedFile.file.fileName);
  if (newName) {
    const updatedFolders = folders.map((folder) =>
      folder.fileName === selectedFile.file.fileName
        ? { ...folder, fileName: newName }
        : folder
    );

    updateDoc(databaseRef, {
      fileLink: updatedFolders,
    });
  }
  closeContextMenu();
};

const handleDelete = () => {
  const updatedFolders = folders.filter(
    (folder) => folder.fileName !== selectedFile.file.fileName
  );

  updateDoc(databaseRef, {
    fileLink: updatedFolders,
  });

  closeContextMenu();
};

const getFile = (event) => {
  const fileRef = ref(storage, event.target.files[0].name);
  const uploadTask = uploadBytesResumable(fileRef, event.target.files[0]);

  uploadTask.on(
    "state_changed",

```

```

(snapshot) => {
  const progress =
    (snapshot.bytesTransferred / snapshot.totalBytes) * 100;

  setProgress(Math.round(progress));
  // eslint-disable-next-line
  switch (snapshot.state) {
    case "paused":
      console.log("Upload is paused");
      break;
    case "running":
      console.log("Upload is running");
      break;
    // eslint-disable-next-line
  }
},
(error) => {},
() => {
  getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {
    updateDoc(databaseRef, {
      fileLink: [
        ...folders,
        {
          downloadURL: downloadURL,
          fileName: event.target.files[0].name,
        },
      ],
    });
  });
});
);
};

const readData = () => {
  onSnapshot(databaseRef, (doc) => {
    setFolders(doc.data().fileLink);
    setFolderName(doc.data().folderName);
  });
};

const openFile = (downloadURL) => {
  window.open(downloadURL, "_blank");
};

```

```

const goBack = () => {
  navigate("/drive");
};

useEffect(() => {
  readData();
}, []); // eslint-disable-line react-hooks/exhaustive-deps

return (
  <
    <div className="back-icon">
      <FiArrowLeftCircle size={50} onClick={goBack} />
    </div>
    <div className="icon-container">
      <div className="upload-btn-wrapper">
        <FiFilePlus color="#18F2B2" className="icon" size={50} />

        <input type="file" onChange={getFile} className="myfile" />
      </div>
    </div>
    <div className="folder-title">
      <h1>{folderName}</h1>
    </div>
    {progress ? <ProgressBar completed={progress} /> : <</>}

    <div className="grid-parent">
      {folders?.map((folder) => {
        return (
          <
            {folder.downloadURL !== "" ? (
              <div
                key={folder.fileName}
                className="preview-child"
                onContextMenu={(event) => openContextMenu(event, folder)}
                onClick={() => openFile(folder.downloadURL)}
              >
                <img
                  className="image-preview"
                  src={folder.downloadURL}
                  alt=""
                />
                <h5>{folder.fileName}</h5>
              </div>
            ) : (

```

```

        ""
    })
</>
);
}}
</div>

{showContextMenu && (
    <div
        className="context-menu"
        style={{
            position: "absolute",
            left: selectedFile.position.x,
            top: selectedFile.position.y,
        }}
        onClick={closeContextMenu}
    >
        <div className="context-menu-option" onClick={handleRename}>
            Rename
        </div>
        <div className="context-menu-option" onClick={handleDelete}>
            Delete
        </div>
    </div>
    )}
</>
);
}

```