ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός – Δημιουργία ιατρικής διαγνωστικής εφαρμογής με την χρήση των γλωσσών Java και Prolog. Συγκριτική ανάλυση.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

των

ΤΣΑΡΟΥΧΑ ΒΑΣΙΛΕΙΟΥ

(ΑΕΜ: 2463)

ΛΟΥΚΟΥΜΗ ΙΩΑΝΝΑΣ

(ΑΕΜ: 2872)

Επιβλέπων : Δόσης Μιχαήλ

Καστοριά 05 - 04 - 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός – Δημιουργία ιατρικής διαγνωστικής εφαρμογής με την χρήση των γλωσσών Java και Prolog. Συγκριτική ανάλυση.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

των

ΤΣΑΡΟΥΧΑ ΒΑΣΙΛΕΙΟΥ

(ΑΕΜ: 2463)

ΛΟΥΚΟΥΜΗ ΙΩΑΝΝΑΣ

(ΑΕΜ: 2872)

Επιβλέπων : Δόσης Μιχαήλ
Ιδιότητα : Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5/04/2023

.....	.....	.....
Δόσης Μιχαήλ	Βέργαδος Δημήτριος	Βαρδάκας Ιωάννης
Καθηγητής	Επίκουρος Καθηγητής	Καθηγητής

Καστοριά 05 - 04 - 2023

ΕΥΧΑΡΙΣΤΙΕΣ

Πρωτίστως θα θέλαμε να εκφράσουμε την ειλικρινή μας ευγνωμοσύνη στον καθηγητή μας κ. Δόση Μιχαήλ, για την συνεργασία, την υποστήριξη και την πολύτιμη συμβολή του. Επίσης και τον καθηγητή μας κ. Μπάτο Παναγιώτη για την ενθάρυνση του καθ' όλη την διάρκεια της πτυχιακής μας εργασίας. Η ακλόνητη πίστη του στις ικανότητες μας, η ανεκτίμητη καθοδήγηση του και οι πολύτιμες συμβουλές του μας βοήθησαν να επιτύχουμε αυτό το ορόσημο στο ακαδημαϊκό μας ταξίδι.

Είμαστε ευγνώμων για τον χρόνο που αφιερώσατε για να μας βοηθήσετε. Τα εποικοδομητικά σχόλια και οι προτάσεις σας μας βοήθησαν να βελτιώσουμε τις ιδέες μας και την συνολική ποιότητα της πτυχιακής μας εργασίας. . Ελπίζουμε να μείνουμε σε επαφή μαζί σας και να συνεχίσουμε να μαθαίνουμε από εσάς στο μέλλον.

Τέλος, θα θέλαμε να εκφράσουμε τις ευχαριστίες μας και στις οικογένειές μας. Χωρίς την υποστήριξη και την ενθάρρυνση τους αυτό το έργο δε θα ήταν δυνατό. Ο συνεχής αγώνας τους για ένα καλύτερο μέλλον για εμάς, μας έδινε ακόμα περισσότερα κίνητρα για να ολοκληρώσουμε την παρούσα πτυχιακή εργασία. Σας ευχαριστούμε για όλα όσα έχετε κάνει για εμάς αλλά και για αυτά που συνεχίζεται να κάνετε.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία εστιάζει στον σχεδιασμό και την υλοποίηση μίας ιατρικής, διαγνωστικής εφαρμογής. Με την χρήση δύο διαφορετικών γλωσσών προγραμματισμού, την Visual Prolog και την Java. Η εφαρμογή έχει υλοποιηθεί στο περιβάλλον ανάπτυξης NetBeans για την Java και σε Visual Prolog 10 IDE για την Visual Prolog. Αρχικά γίνεται έρευνα για τα ιατρικά διαγνωστικά προγράμματα αλλά και τις εφαρμογές που υπάρχουν ήδη. Με σκοπό την εξοικείωση και την άντληση πληροφοριών. Έπειτα γίνεται παρουσίαση των εργαλείων ανάπτυξης, σχεδιασμού και διαγραμμάτων UML και ER. Στη συνέχεια γίνεται ανάλυση του κώδικα της εφαρμογής και στις δύο γλώσσες. Όπως και ανάλυση της φιλικότητας προς τον χρήστη κατά τη χρήση της εφαρμογής. Ωστόσο κατά την παρουσίαση των δύο γλωσσών και της εφαρμογής που υλοποιήθηκε σε αυτες, παρατηρούνται αρκετές διαφορές. Στα πλαίσια της συγκριτικής ανάλυσης, παρατίθενται και αναπτύσσοσνται περεταίρω αυτες οι διαφορές μεταξύ των δύο γλωσσών προγραμματισμού, αλλά και οι δυαφορες που δημιουργήθηκαν κατα την σχεδίαση και την υλοποίση της εφαρμογής.

Λέξεις κλειδία : Εφαρμογή, διαγνωστικό πρόγραμμα, περιβάλλον ανάπτυξης, Visual Prolog, Java, UML, ER, NetBeans, Visual Prolog 10 IDE

ABSTRACT

This thesis focuses on the design and implementation of a medical, diagnostic application, by using two different programming languages, Visual Prolog and Java. The application has been implemented in the NetBeans development environment for Java and in Visual Prolog 10 IDE for Visual Prolog. Initially, research is done on the medical diagnostic programs as well as the applications that already exist. For the purpose of familiarization and obtaining information. Then, the design of the development tools is presented, together with the UML and ER diagrams. The application code is then analyzed in both languages. As well as analysis of user-friendliness while using the application. However, when presenting the two languages and the application implemented in them, several differences are observed. In the context of the comparative analysis, these differences between the two programming languages, as well as the differences created during the design and implementation of the application, are presented, and further developed.

Keywords : Application, diagnostic program, development environment, Visual Prolog, Java, UML, ER, NetBeans, Visual Prolog 10 IDE

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

Εισαγωγή

Η ιατρική διάγνωση είναι μια κρίσιμη πτυχή της υγειονομικής περίθαλψης που μπορεί να είναι προκλητική και χρονοβόρα, ιδιαίτερα όταν βασίζεται αποκλειστικά στην τεχνογνωσία των επαγγελματιών υγείας. Η χρήση διαγνωστικών προγραμμάτων και εργαλείων που βασίζονται σε υπολογιστή μπορεί να βοηθήσει τους παρόχους υγειονομικής περίθαλψης να κάνουν ακριβείς και έγκαιρες διαγνώσεις. Η παρούσα πτυχιακή εργασία εστιάζει στον σχεδιασμό και την υλοποίηση μίας διαγνωστικής ιατρικής εφαρμογής, με την χρήση των γλωσσών προγραμματισμού Java και Visual Prolog.

Συγκεκριμένα και με βάση την σειρά δημιουργίας της παρούσης εργασίας, στο πρώτο κεφάλαιο αναφέρονται οι λειτουργικοί ορισμοί, τα ερευνητικά δεδομένα και η θεωρία της Java και της Visual Prolog. Έπειτα στο δεύτερο κεφάλαιο αναφέρονται και αναλύονται τα προγράμματα που χρησιμοποιήθηκαν για την κατασκευή της εφαρμογής. Τα οποία είναι το NetBeans και το Visual Prolog 10 IDE, για την Java και την Visual Prolog αντίστοιχα. Καθώς και η βάση δεδομένων SQLite που ενσωματώθηκε στην Java, για την αποθήκευση των στοιχείων και του αποτελέσματος της διάγνωσης του ασθενή. Επίσης δημιουργούνται και επεξηγούνται τα διαγράμματα UML και ER. Αυτά τα δεδομένα και στοιχεία, έχουν ως σκοπό ο αναγνώστης – χρήστης να κατανοήσει την χρησιμότητα, την λειτουργικότητα και τον στόχο της εφαρμογής. Αλλά και να μάθει κάποιους βασικούς ορισμούς που θα αξιοποιηθούν παρακάτω στην δημιουργία και την παρουσίαση της.

Κατά την δημιουργία της εφαρμογής χρησιμοποιήθηκαν αρκετοί κανόνες και στην Java αλλά και στην Visual Prolog. Για την καλύτερη σχεδίαση του γραφικού περιβάλλοντος και την λειτουργικότητα της εφαρμογής. Όμως, κατά την υλοποίηση της εφαρμογής προέκυψαν αρκετές διαφορές μεταξύ των δύο γλωσσών. Οι δύο γλώσσες είναι ούτως ή άλλως αρκετά διαφορετικές μεταξύ τους. Η Java είναι αντικειμενοστραφής γλώσσα προγραμματισμού, ενώ η Visual Prolog είναι μια γλώσσα λογικού προγραμματισμού με στοιχεία αντικειμενοστραφούς. Στην τελευταία ενότητα της πτυχιακής παρατίθενται οι διαφορές τους.

Εν κατακλείδι, τα ευρήματα αυτής της έρευνας θα μπορούσαν να συμβάλουν στην ανάπτυξη πιο αποτελεσματικών και αποδοτικών ιατρικών διαγνωστικών εφαρμογών, προς όφελος τόσο των παρόχων υγειονομικής περίθαλψης όσο και των ασθενών.

Κεφάλαιο 1ο

Το πρώτο κεφάλαιο της παρούσης εργασίας αναφέρεται στο διαγνωστικό πρόγραμμα, σε γενικό ορισμό αλλά και στην χρήση του στα ιατρικά εργαλεία. Αναφέρει τις ασθένειες που χρησιμοποιεί η εφαρμογή ως δεδομένα για την υλοποίηση της. Τις γλώσσες προγραμματισμού Java και Visual Prolog, όπως αυτές αναφέρονται και αναλύονται. Τόσο από τους ίδιους τους δημιουργούς, όσο και από τους προγραμματιστές που τις χειρίζονται. Με τα πλεονεκτήματα και τα μειονεκτήματά τους. Επιπλέον, παρουσιάζεται και ένα παράδειγμα κώδικα οικογενειακού δέντρου, μαζί με το αποτέλεσμά του και στις δύο γλώσσες. Τέλος θα αναφερθούμε σε μερικές εφαρμογές που υπάρχουν στην αγορά, παρόμοιες με αυτήν που σχεδιάζεται και δημιουργείτε στα επόμενα κεφάλαια.

1.1 Διαγνωστικά προγράμματα

Οι διαδικαστικές οδηγίες που εκτελούσαν οι άνθρωποι στο χώρο της εργασίας τους, μπορούν να θεωρηθούν ως η αρχή των διαγνωστικών προγραμμάτων. Τη δεκαετία του 1960 η εταιρεία αυτοκινήτων Volkswagen εισήγαγε το πρώτο σύστημα υπολογιστή, σε όχημα με δυνατότητες διαγνωστικού ελέγχου. [1]

Σήμερα με τον όρο διαγνωστικό πρόγραμμα (Test Mode) εννοούμε την αυτόματη ακολουθία ενός προγράμματος του υπολογιστή. Το οποίο προσδιορίζει με ακρίβεια τη λειτουργική κατάσταση του λογισμικού, του υλικού ή οποιονδήποτε συνδυασμό αυτών σε ένα εξάρτημα, σε ένα σύστημα ή σε ένα δίκτυο συστημάτων. Ένα τέτοιο πρόγραμμα, ιδανικά παρέχει στους χηριστές του οδηγίες για πιθανά θέματα ή σφάλματα που μπορεί να προκύψουν κατά τη λειτουργία.

Επιπλέον μπορεί να είναι απλά ή πιο σύνθετα προγράμματα. Να λειτουργούν απρόοπτα σε συσκευές που χρησιμοποιούνται καθημερινά ή να περιμένουν την ενεργοποίηση τους για να προβούν σε πιο σύνθετες εκτιμήσεις αποδόσεων. Το διαγνωστικό πρόγραμμα για μία συσκευή ή ένα σύστημα, μπορεί να είναι εγκατεστημένο σε ανεξάρτητη θέση ή να είναι ενσωματωμένο σε αυτό.

Παρακάτω παρατίθενται οι μέθοδοι λειτουργίας τους :

- Η παρακολούθηση των δεικτών του συστήματος, η οποία περιλαμβάνει στατιστική ανάλυση των τάσεων και καταγραφή μη φυσιολογικών γεγονότων.
- Τα διαγνωστικά που βασίζονται σε λύσεις, τα οποία ελέγχουν για γνωστούς τρόπους αποτυχίας ανιχνεύοντας τα συμπτώματά τους.

- Η δοκιμή μαύρου κουτιού, η οποία εστιάζει στην ακρίβεια των δεδομένων εξόδου με βάση μια γνωστή είσοδο χωρίς γνώση του πώς λειτουργεί ο μηχανισμός.
- Η δοκιμή λευκού κουτιού, η οποία ελέγχει άμεσα τις εσωτερικές λειτουργίες ενός μηχανισμού χρησιμοποιώντας τη γνώση του πώς λειτουργεί.
- Η προσανατολισμένη στη λειτουργία μέθοδος, η οποία συνδυάζει τη δοκιμή μαύρου και λευκού κουτιού παρεμβάλλοντας μία ή περισσότερες λειτουργίες μαύρου κουτιού με μία ή περισσότερες λειτουργίες λευκού κουτιού. Αν και δεν προτιμώνται, ορισμένα πολύπλοκα συστήματα μπορεί να απαιτούν αυτόν τον τύπο δοκιμών επειδή δεν διαθέτουν τις απαραίτητες διεπαφές για να εκτελούν τον ένα ή τον άλλο τύπο ανεξάρτητα.
- Η ενσωματωμένη διάγνωση στο παρασκήνιο, η οποία εκτελεί δοκιμές των στοιχείων του συστήματος κατά τη διάρκεια του χρόνου αδράνειας ενός συστήματος.
- Η λειτουργία διαπλεκόμενα διαγνωστικά (interleaved diagnostics), η οποία ενσωματώνει τη διάγνωση στην κανονική λειτουργία ενός στοιχείου του συστήματος για την άμεση διάγνωση οποιουδήποτε οριακού τρόπου λειτουργίας. Για παράδειγμα στοιχεία υλικού με δυνατότητες που βοηθούν ένα διαγνωστικό πρόγραμμα περιλαμβάνουν σύγχρονους σκληρούς δίσκους με εντολές Self-Monitoring, Analysis and Reporting Technology (SMART) που παρέχουν πληροφορίες σχετικά με συνθήκες εσωτερικών σφαλμάτων, όπως μετρήσεις επανάληψης περιστροφής και μετρήσεις κακών τομέων. Ορισμένα συστήματα ενδέχεται επίσης να χρησιμοποιούν μνήμη κωδικού διόρθωσης σφάλματος (ECC), που καταγράφει συμβάντα αποτυχίας μνήμης που διορθώθηκαν αυτόματα.

Επιπλέον υπάρχουν και διάφοροι τύποι διαγνωστικών προγραμμάτων που χρησιμοποιούνται για διάφορους σκοπούς :

- Διαγνωστικά ενός σκοπού, τα οποία είναι προγράμματα που έχουν συγκεκριμένο σκοπό και χρησιμοποιούνται για την επικύρωση ή τον έλεγχο συγκεκριμένων διαμορφώσεων. Για παράδειγμα, ένα πρόγραμμα που επικυρώνει τη διαμόρφωση του DirectX των Windows.
- Διαγνωστικά πολλαπλού σκοπού, τα οποία είναι μονολιθικά προγράμματα που εκτελούν πολλαπλές εργασίες. Αυτά μπορεί να μην είναι κατάλληλα για όλες τις χρήσεις, καθώς είναι παρόμοια με ένα σφυρί που λειτουργεί καλά με καρφιά αλλά κακώς με βίδες και παξιμάδια.

- Αρθρωτά διαγνωστικά, τα οποία αποτελούνται από σύνολα διαγνωστικών ενός σκοπού που συνδυάζονται σε ένα περιβάλλον που μπορεί να προσαρμοστεί ώστε να ανταποκρίνεται σε συγκεκριμένες απαιτήσεις του κλάδου. Αυτά τα διαγνωστικά διαθέτουν ένα επαναχρησιμοποιήσιμο λειτουργικό σύστημα υλικού και λογισμικού που εκτελεί όλα τα προγράμματά τους. Παραδείγματος χάρη περιλαμβάνουν δοκιμές κατασκευής που ελέγχουν θέματα που σχετίζονται με τη συναρμολόγηση και βελτιστοποιούν το χρόνο, διαγνωστικά που απευθύνονται στον τελικό χρήστη, τα οποία έχουν μια εύληπτη παρουσίαση και επικεντρώνονται σε λύσεις, δοκιμές σέρβις/εγγύησης που εντοπίζουν αποτυχημένες ή οριακές μονάδες αντικατάστασης πεδίου (FRU) και διαγνωστικά που επικεντρώνονται στην ανακαίνιση και καθορίζουν αν ένα σύστημα μπορεί να μεταπωληθεί ή να επαναχρησιμοποιηθεί με κόστος στο χρόνο που δαπανάται για τις δοκιμές.
- Διαγνωστικά συστήματα βασισμένα στη γνώση, τα οποία βασίζονται στη γνώση και την εμπειρία των τεχνικών ή των διαγνωστών. Το σύστημα χρησιμοποιεί ένα "νοητικό μοντέλο" της λειτουργίας του συστήματος και λογικούς συλλογισμούς για να εντοπίσει μία ή περισσότερες πιθανές αιτίες για την ύπαρξη μιας κατάστασης.

1.1.1 Ιατρική και διαγνωστικό πρόγραμμα

Ένα ιατρικό διαγνωστικό εργαλείο (λογισμικό) είναι ένας τύπος προγράμματος ή εφαρμογής υπολογιστή που έχει σχεδιαστεί για να βοηθά τους επαγγελματίες υγείας να διαγνώσουν ιατρικές καταστάσεις ή ασθένειες. Αυτά τα εργαλεία χρησιμοποιούν συνήθως αλγορίθμους και βάσεις δεδομένων ιατρικών γνώσεων για να αναλύουν τα δεδομένα και τα συμπτώματα των ασθενών και να παρέχουν διαγνωστικές προτάσεις ή συστάσεις. Υπάρχουν πολλοί διαφορετικοί τύποι ιατρικών διαγνωστικών εργαλείων, που κυμαίνονται από απλούς ελεγκτές συμπτωμάτων έως πιο σύνθετα συστήματα υποστήριξης αποφάσεων. [2]

Ορισμένα κοινά παραδείγματα παρουσιάζονται παρακάτω.

- Συστήματα ηλεκτρονικού φακέλου υγείας (EHR): Πρόκειται για προγράμματα λογισμικού που συλλέγουν και αποθηκεύουν πληροφορίες για την υγεία των ασθενών, συμπεριλαμβανομένου του ιατρικού ιστορικού, των αποτελεσμάτων των εξετάσεων και των σχεδίων θεραπείας. Τα συστήματα EHR μπορούν να χρησιμοποιηθούν για την υποστήριξη της κλινικής λήψης αποφάσεων και να

βοηθήσουν τους παρόχους υγειονομικής περίθαλψης να διαγνώσουν και να θεραπεύσουν τους ασθενείς.

- Συστήματα υποστήριξης κλινικών αποφάσεων (CDSS): Πρόκειται για προγράμματα υπολογιστών που χρησιμοποιούν αλγόριθμους και ιατρικές γνώσεις για την παροχή διαγνωστικών και θεραπευτικών συστάσεων με βάση τα δεδομένα του ασθενούς. Τα εργαλεία CDSS μπορούν να χρησιμοποιηθούν για να υποστηρίξουν τους παρόχους υγειονομικής περίθαλψης στη λήψη ακριβών και έγκαιρων διαγνώσεων, καθώς και στην ανάπτυξη αποτελεσματικών σχεδίων θεραπείας.
- Λογισμικό ιατρικής απεικόνισης: Αυτά τα εργαλεία χρησιμοποιούν τεχνολογίες ιατρικής απεικόνισης, όπως ακτίνες Χ, μαγνητικές τομογραφίες και αξονικές τομογραφίες, για την παραγωγή λεπτομερών εικόνων του εσωτερικού του σώματος. Το λογισμικό ιατρικής απεικόνισης μπορεί να χρησιμοποιηθεί για να βοηθήσει στη διάγνωση και την παρακολούθηση ενός ευρέος φάσματος ιατρικών καταστάσεων, από σπασμένα οστά έως καρκίνο.
- Συστήματα διαχείρισης εργαστηριακών πληροφοριών (LIMS): Πρόκειται για προγράμματα λογισμικού που χρησιμοποιούνται για τη διαχείριση εργαστηριακών λειτουργιών, συμπεριλαμβανομένης της παρακολούθησης δειγμάτων, της ανάλυσης δεδομένων και της υποβολής εκθέσεων. Τα εργαλεία LIMS μπορούν να χρησιμοποιηθούν για να βοηθήσουν στη διάγνωση ιατρικών καταστάσεων με την ανάλυση αίματος, ούρων και άλλων σωματικών υγρών.

Συνολικά, τα ιατρικά διαγνωστικά εργαλεία διαδραματίζουν σημαντικό ρόλο στο να βοηθούν τους παρόχους υγειονομικής περίθαλψης να κάνουν ακριβείς και έγκαιρες διαγνώσεις, οι οποίες μπορούν να οδηγήσουν σε αποτελεσματικότερες θεραπείες και καλύτερα αποτελέσματα για τους ασθενείς.

1.2 Οι ασθένειες της εφαρμογής

Η ασθένεια είναι μια ιατρική κατάσταση ή διαταραχή που επηρεάζει τη φυσιολογική λειτουργία του οργανισμού. Οι ασθένειες μπορούν να προκληθούν από διάφορους παράγοντες, όπως ιούς, βακτήρια, γενετικές μεταλλάξεις, περιβαλλοντικούς παράγοντες και επιλογές του τρόπου ζωής. Ορισμένα κοινά παραδείγματα ασθενειών περιλαμβάνουν τον διαβήτη, τις καρδιακές παθήσεις, τον καρκίνο και τις μολυσματικές ασθένειες όπως τη γρίπη και τον COVID-19.

Τα συμπτώματα είναι τα σημάδια ή οι ενδείξεις μιας ασθένειας ή μιας ιατρικής κατάστασης. Μπορεί να είναι σωματικά, όπως πόνος, πυρετός ή εξάνθημα, ή μπορεί να είναι ψυχολογικά, όπως άγχος ή κατάθλιψη. Επίσης, μπορεί να ποικίλλουν ανάλογα με τον τύπο και τη σοβαρότητα της νόσου και διαφορετικοί άνθρωποι μπορεί να εμφανίζουν διαφορετικά συμπτώματα για την ίδια ασθένεια.

Προκειμένου να διαγνώσουν μια ασθένεια, οι γιατροί χρησιμοποιούν συνήθως έναν συνδυασμό ιατρικού ιστορικού, φυσικής εξέτασης και διαγνωστικών εξετάσεων. Ο στόχος ενός ιατρικού διαγνωστικού εργαλείου είναι να βοηθήσει τους γιατρούς να προσδιορίσουν μια ασθένεια ή πάθηση με βάση τα συμπτώματα του ασθενούς και άλλες σχετικές πληροφορίες. Αυτό μπορεί να επιτευχθεί με τη χρήση διαγνωστικών εφαρμογών, αλγορίθμων, τεχνητής νοημοσύνης και άλλων τεχνολογιών που αναλύουν μεγάλες ποσότητες ιατρικών δεδομένων για τον εντοπισμό μοτίβων και συσχετίσεων μεταξύ συμπτωμάτων και ασθενειών.

Με την βοήθεια διαγνωστικών ιατρικών εφαρμογών συλλέγονται δεδομένα συμπτωμάτων και άλλων πληροφοριών του ασθενούς. Τα ιατρικά διαγνωστικά εργαλεία των εφαρμογών μπορούν να βοηθήσουν τους γιατρούς να κάνουν διαγνώσεις και τους χρήστες των εφαρμογών να μάθουν άμεσα και γρηγόρα με μία πρώτη διάγνωση από τι πάσχουν.

Στην παρούσα πτυχιακή εργασία σχεδιάζεται και δημιουργείται μια διαγνωστική εφαρμογή. Στην οποία τα δεδομένα των συμπτωμάτων των ασθενείων που έχουν καταχωρηθεί στη βάση δεδομένων, έχουν συλεχθεί από επίσημες ιατρικές σελιδές και παρατίθενται αναλυτικά στις παρακάτω υποενότητες.

1.2.1 COVID-19

Ο COVID-19 έχει κάνει την εμφάνιση του παγκοσμίος τα τελευταία 4 χρόνια. Ο Ιος που προκαλεί αυτη την ασθένεια είναι ο SARS-COV-2. Μελέτες έχουν δείξει πως τα συμπτώματα του είναι ποικίλα. Αυτό οφείλεται στις διάφορες μεταλλάξεις του. Αντιδράει και αναπαράγεται ανάλογα με τις καιρικές θερμοκρασίες. Διότι ο ιος αντέχει στο κρύο και είναι πιο μεταδοτικός με αποτέλεσμα να προκαλέι περίσσοτερα συμπτώματα στους ασθενείς. Ενώ το καλοκαιρι εξασθενεί και έχει ως αποτέλεσμα τα συμπτώματά του να είναι πιο ήπια και με πιο χαμηλή μεταδοτικότητα. Για να εμφανιστούν στον οργανισμό του ατόμου που έχει εκτεθεί στον ιο, μπορεί να κάνουν από δύο εώς και 14 μέρες. Οι άνθρωποι που πάσχουν από χρόνια, βαριά υποκείμενα νοσήματα αλλά και οι ηλικιωμένοι διατρέχουν μεγαλύτερο κίνδυνο, να νοσήσουν και να αισθάνονται σε μέγιστο βαθμό τα συμπτώματα που θα τους προκαλέσει ο ιος. Τα πιο πιθανά συμπτώματα και ανεξαρτήτος την μετάλλαξη του είναι ο πυρετός, ο βήχας, η τάση προς

έμετο, η διάρροια, η δυσκολία στην αναπνοή, οι πόνοι στους μυες ή σε όλο το σώμα, η απώλεια γεύσης ή όσφρησης και άλλα. [3]

1.2.2 Φαρυγγίτιδα

Η φαρυγγίτιδα είναι στην ουσία λοίμωξη στον στοματοφάρυγγα. Είναι άμεσα μεταδοτική από άτομο σε άτομο. Η μετάδοσή της γίνεται μέσω σταγονιδίων του αναπνευστικού, από σάλιο ή και από ρινικές εκκρίσεις του ατόμου που νοσεί. Τα σχολεία, οι παιδικοί σταθμοί ακόμα και οι στρατιωτικές εγκαταστάσεις είναι μέρη όπου διευκολύνουν την μετάδοσή της, εξαιτίας της πολυκοσμίας και του συνοστισμού που επικρατεί σε αυτά τα μέρη. Οι γιατροί στα άτομα που έχουν προσβληθεί από φαρυγγίτιδα, συνταγογραφούν άμεσα αντιβίωση και τους συμβουλεύουν να παραμείνουν σπίτι τους από 2 εώς και 5 μέρες. Τα συμπτώματα της είναι η ξαφνική έναρξη πονόλαιμου, ο πόνος κατά την κατάποση, ο πυρετός, ο βήχας, η ρινόρροια, η βραχνάδα, η στοματική έλκη και η φλόγωση της μεμβράνης των βλεφάρων. [4]

1.2.3 Πνευμονία

Πνευμονία ονομάζεται η λοίμωξη των πνευμόνων. Είναι μία πολύ σοβαρή ασθένεια, η οποία έχει ποσοστό ένας στους είκοσι που νοσεί, να οδηγηθεί σε θάνατο. Μερικές επιπλοκές της είναι το εμπύημα, η περικαρδίτιδα και η ενδοβρογχική απόφραξη με ατελκασία και απόστημα στους πνεύμονες. Τα πιο πιθανά συμπτώματα της πνευμονίας είναι ο πυρετός, η ρίγη, ο βήχας, η ταχεία αναπνοή ή η δυσκολία στην αναπνοή και ο πόνος στο στήθος. Μερικές φορές τα συμπτώματά της είναι και άλλα και εξαρτώνται κυρίως απο τον βαθμό που νοσεί, το ιστορικό και την ηλίκια του ασθενεί. [5]

1.2.4. Μηνιγγίτιδα

Η μηνιγγίτιδα είναι η λοίμωξη της επένδυσης του εγκεφάλου και του νωτιαίου μυελού. Είναι μία πολύ σοβαρή και επικίνδυνη ασθένεια. Έχει ποσοστό ένα στα δώδεκα παιδιά και ένας στους έξι ενήλικες που νοσούν να πεθαίνουν από την μόλυνση. Όμως οι περισσότεροι που θα αναρώσουν από την μηνιγγίτιδα θα αποκτήσουν μακροχρόνια προβλήματα. Μερικά από αυτά τα προβλήματα είναι η απώλεια ακοής, κυρίως στους μεγαλύτερους ηλικιακά ανρθώπους και η αναπτυξιακή καθυστέρηση σε άτομα νεαρής ηλικίας. Τα συμπτώματα της είναι ο πυρετός, ο πονοκέφαλος, η φωτοφοβία, η σύγχυση και το στραβολαίμιασμα. [6]

1.2.5 Κοινό Κρυολόγημα

Το κοινό κρυολόγημα μπορεί να προσβληθεί στο άτομο από διάφορους, διαφορετικούς ιούς. Μερικοί από αυτούς τους ιούς είναι ο εποχιακός κοροναϊός, η παραγρίπη και ο ρινοϊός. Τα συμπτώματα του κρυολογήματος είναι πολύ ήπια και δεν προκαλούν κάποιο σοβαρό πρόβλημα στον οργανισμό του νοσήσαντα. Τα συμπτώματά του είναι η καταρροή, η βουλωμένη μύτη, ο πονόλαιμος και ο πονοκέφαλος. Αν και πολλοί μπερδεύουν το κοινό κρυολόγημα με την γρίπη. Διότι έχουν κατά βάση τα ίδια συμπτώματα. [7]

1.2.6 Γρίπη

Η Γρίπη είναι μία αρκετά μεταδοτική ασθένεια που προκαλείται από ιούς γρίπης. Συνήθως, είναι ήπια σε μορφή και μπορούν να προσβληθούν από αυτή όλες οι ηλικίες. Ωστόσο τα συμπτώματά της είναι έντομα και εμφανίζονται απότομα. Επομένως, μερικές φορές και για κάποιους οργανισμούς μπορεί να θεωρηθεί μία πολύ σοβαρή ασθένεια, η οποία έχει την δυνατότητα να προκαλέσει μέχρι και τον θάνατο του ασθενή. Τα βασικά συμπτώματα της γρίπης είναι ο πυρετός, ο βήχας, ο πονόλαιμος, η καταρροή ή, η βουλωμένη μύτη, οι πόνοι στους μύες ή στο σώμα, ο πονοκέφαλος και η κόπωση. [8]

1.3 Εργαλεία

Τα εργαλεία μπορούν να οριστούν ως όργανα, συσκευές ή εφαρμογές λογισμικού που έχουν σχεδιαστεί για να βοηθούν στην ολοκλήρωση μιας συγκεκριμένης εργασίας ή ενός στόχου. Συχνά δημιουργούνται για τη βελτίωση της αποτελεσματικότητας, της ακρίβειας ή της ευκολίας χρήσης και χρησιμοποιούνται σε ένα ευρύ φάσμα βιομηχανιών και κλάδων.

Τα φυσικά εργαλεία μπορεί να περιλαμβάνουν εργαλεία χειρός, όπως σφυριά, κατσαβίδια και κλειδιά, καθώς και πιο εξειδικευμένο εξοπλισμό, όπως πριόνια, τρυπάνια και τόρνους. Αυτά τα εργαλεία χρησιμοποιούνται στις κατασκευές, τη μεταποίηση και άλλες βιομηχανίες για την εκτέλεση συγκεκριμένων εργασιών, όπως η κοπή υλικών ή η συναρμολόγηση εξαρτημάτων.

Εκτός από τα φυσικά εργαλεία, τα ψηφιακά εργαλεία έχουν γίνει όλο και πιο σημαντικά στον σημερινό κόσμο. Τα εργαλεία αυτά περιλαμβάνουν εφαρμογές λογισμικού που χρησιμοποιούνται σε διάφορους τομείς, όπως οι επεξεργαστές κειμένου και τα λογιστικά

φύλλα που χρησιμοποιούνται στις επιχειρήσεις ή το λογισμικό σχεδιασμού και προγραμματισμού που χρησιμοποιείται στην τεχνολογία και τις δημιουργικές βιομηχανίες.

Τα ψηφιακά εργαλεία μπορούν να βοηθήσουν στην αυτοματοποίηση επαναλαμβανόμενων εργασιών, να παρέχουν πρόσβαση σε τεράστιες ποσότητες πληροφοριών ή να καταστήσουν δυνατή την πολύπλοκη ανάλυση δεδομένων. Μπορούν επίσης να χρησιμοποιηθούν για την επικοινωνία και τη συνεργασία, επιτρέποντας σε άτομα ή ομάδες να συνεργαστούν εξ αποστάσεως σε ένα κοινό έργο.

Συνολικά, τα εργαλεία αποτελούν κρίσιμο μέρος της ανθρώπινης προόδου και της καινοτομίας, καθώς βοηθούν τα άτομα και τους οργανισμούς να επιτελέσουν εργασίες που διαφορετικά θα ήταν δύσκολο ή αδύνατο να επιτευχθούν.

Στις παρακάτω υποενότητες θα δούμε για τα εργαλεία προγραμματισμού. Στο αρχικό τους στάδιο, το οποίο είναι οι γλώσσες προγραμματισμού. Πιο συγκεκριμένα θα γίνει λόγος για τις γλώσσες προγραμματισμού Java και Visual Prolog. Όπως αυτές ορίζονται από ένα σύνολο συντακτικών και εννοιολογικών κανόνων, που ορίζουν την δομή και το νόημα, αντίστοιχα των προτάσεων της γλώσσας.

1.3.1 Visual Prolog

Η Visual Prolog είναι μια γλώσσα προγραμματισμού και ένα περιβάλλον ανάπτυξης που αναπτύχθηκε αρχικά από τη Prolog Development Center (PDC) στη Δανία. Βασίζεται στη λογική γλώσσα προγραμματισμού Prolog, αλλά με επεκτάσεις για υποστήριξη αντικειμενοστρεφούς προγραμματισμού και γραφική διεπαφή χρήστη. Έχει σχεδιαστεί για να είναι ένα αποτελεσματικό και ισχυρό εργαλείο για την ανάπτυξη πολύπλοκων εφαρμογών όπως έμπειρα συστήματα, επεξεργασία φυσικής γλώσσας και εφαρμογές τεχνητής νοημοσύνης. [9],[10]

Είναι μια γλώσσα λογικού προγραμματισμού που βασίζεται στις έννοιες του κατηγορηματικού λογισμού και της λογικής πρώτης τάξης. Δηλαδη είναι μια γλώσσα προγραμματισμού και ένα περιβάλλον ανάπτυξης για τη δημιουργία έξυπνων εφαρμογών, που βασίζεται στις αρχές της λογικής και της αφαίρεσης, με τα προγραμματα να αποτελούνται από μια σειρά γεγονότων και κανόνων που ορίζουν τις σχέσεις μεταξύ των αντικειμένων.

Ένα από τα βασικά χαρακτηριστικά της Visual Prolog είναι η υποστήριξή του για αντικειμενοστραφή προγραμματισμό, ο οποίος επιτρέπει στους προγραμματιστές να

οργανώσουν τον κώδικά τους σε κλάσεις και αντικείμενα. Ετσι, διευκολύνει τη δημιουργία πολύπλοκων εφαρμογών με επαναχρησιμοποιήσιμο κώδικα.

Η Visual Prolog χρησιμοποιείται κυρίως για την ανάπτυξη εξειδικευμένων συστημάτων, εφαρμογών επεξεργασίας φυσικής γλώσσας και άλλων ευφυών συστημάτων. Είναι μια γλώσσα με έντονη πληκτρολόγηση, πράγμα που σημαίνει ότι οι προγραμματιστές πρέπει να δηλώσουν τον τύπο δεδομένων κάθε μεταβλητής πριν τη χρησιμοποιήσουν. Επίσης έχει αυτόματη διαχείριση μνήμης και περιλαμβάνει μια σειρά ενσωματωμένων κατηγορημάτων και συναρτήσεων για εργασία με κοινούς τύπους δεδομένων, όπως ακεραίων, αριθμών κινητής υποδιαστολής, χαρακτήρων, συμβολοσειρών, δυαδικών, λίστες, αντικέιμενα, πίνακες και δομές. Με την ενσωματωμένη βάση δεδομένων που διαθέτει γίνεται πιο εύκολη η αποθήκευση και η ανάκτηση δεδομένων.

Ένα από τα μοναδικά χαρακτηριστικά της Visual Prolog είναι η υποστήριξή της για προγραμματισμό λογικής περιορισμών, ο οποίος επιτρέπει στους προγραμματιστές να καθορίζουν σχέσεις και περιορισμούς μεταξύ διαφορετικών αντικειμένων και δομών δεδομένων. Για παράδειγμα, ένα πρόγραμμα λογικής περιορισμών μπορεί να ορίσει ένα σύνολο κανόνων που περιγράφουν τις σχέσεις μεταξύ διαφορετικών τμημάτων ενός πολύπλοκου συστήματος και στη συνέχεια να χρησιμοποιήσει αυτούς τους κανόνες για να λύσει προβλήματα ή να κάνει προβλέψεις. Αυτό μπορεί να είναι χρήσιμο για την ανάπτυξη πολύπλοκων εφαρμογών που περιλαμβάνουν περίπλοκες σχέσεις και αλληλεξαρτήσεις.

Η Visual Prolog περιλαμβάνει ένα σύνολο βιβλιοθηκών και εργαλείων για τη δημιουργία γραφικών διεπαφών χρήστη (GUI) . Αυτό καθιστά εύκολη τη δημιουργία εφαρμογών με κουμπιά, μενού, πλαίσια κειμένου και άλλα στοιχεία διεπαφής χρήστη. Συνοδεύεται από ένα ενσωματωμένο περιβάλλον ανάπτυξης (IDE) που περιλαμβάνει πρόγραμμα επεξεργασίας κώδικα, μια βιβλιοθήκη με προκατασκευασμένα στοιχεία ελέγχου, πρόγραμμα εντοπισμού σφαλμάτων και άλλα εργαλεία για την ανάπτυξη και τη δοκιμή εφαρμογών. Επιπλέον διαθέτει αντιστοίχιση προτύπων, η οποία επιτρέπει την αντιστοίχιση ενος μοτίβου με ένα σύνολο γεγονότων ή κανόνων. Αυτή η δυνατότητα είναι χρήσιμη για αναζήτηση μέσω βάσεων δεδομένων ή άλλων μεγάλων συνόλων δεδομένων.

Επιπροσθέτως, υποστηρίζει πολλές πλατφόρμες(Cross-Platform), συμπεριλαμβανομένων των Windows, Linux και macOS. Μπορεί να ενσωματωθεί με άλλες γλώσσες όπως η C++ και η Java, επιτρέποντάς τη σύνταξη ενος τμήματος της εφαρμογής σε διαφορετικές γλώσσες και να συνδιαστεί σε μια ενιαία εφαρμογή. Όσον αφορά τη σύνταξη, η Visual Prolog είναι

παρόμοια με αυτή της Prolog, , αλλά με πρόσθετες δυνατότητες για αντικειμενοστραφή προγραμματισμό. Η γλώσσα υποστηρίζει χαρακτηριστικά όπως κληρονομικότητα, πολυμορφισμός και ενθυλάκωση. Με ένα σύνολο λέξεων-κλειδιών, τελεστών και δομών ελέγχου για τον ορισμό της λογικής και των κανόνων. Ωστόσο, η σύνταξη και η δομή της γλώσσας μπορεί να χρειαστεί λίγη εξοικείωση, ειδικά για προγραμματιστές που είναι πιο εξοικειωμένοι με επιτακτική ή αντικειμενοστραφείς γλώσσες προγραμματισμού.

Όπως κάθε άλλη γλώσσα προγραμματισμού ή εργαλείο, η Visual Prolog έχει τα πλεονεκτήματα και τα μειονεκτήματά της. Μερικά από τα μειονεκτήματα της είναι η απότομη καμπύλη εκμάθησης. Ιδιαίτερα για όσους είναι νέοι στον προγραμματισμό, η σύνταξη και η δομή της διαφέρουν αρκετά από άλλες γλώσσες προγραμματισμού, γεγονός που μπορεί να την κάνει δύσκολη στην εκμάθηση. Επίσης ενώ υπάρχει διαθέσιμη μια δωρεάν έκδοση της, η πλήρης έκδοση μπορεί να είναι αρκετά ακριβή. Αυτό μπορεί να την κάνει λιγότερο προσβάσιμη σε μικρότερες επιχειρήσεις ή ιδιώτες.

Σε σύγκριση με άλλες γλώσσες προγραμματισμού και εργαλεία, η Visual Prolog έχει μια σχετικά μικρή κοινότητα χρηστών. Αυτό δυσκολεύει την εύρεση βοήθειας ή πόρων κατά την αντιμετώπιση κάποιου προβλήματος. Διότι χρησιμοποιείται κυρίως για την ανάπτυξη επιτραπέζιων εφαρμογών για Windows. Δεν είναι κατάλληλη για την ανάπτυξη εφαρμογών ιστού ή εφαρμογών για κινητά. δεν έχει τόσες ενσωματώσεις με άλλα εργαλεία και τεχνολογίες όσο άλλες γλώσσες προγραμματισμού. Αυτό μπορεί να καταστήσει πιο δύσκολη τη χρήση του σε ορισμένα περιβάλλοντα, όπως κατά την ανάπτυξη πολύπλοκων συστημάτων λογισμικού που απαιτούν ενσωμάτωση με πολλαπλά εργαλεία και τεχνολογίες.

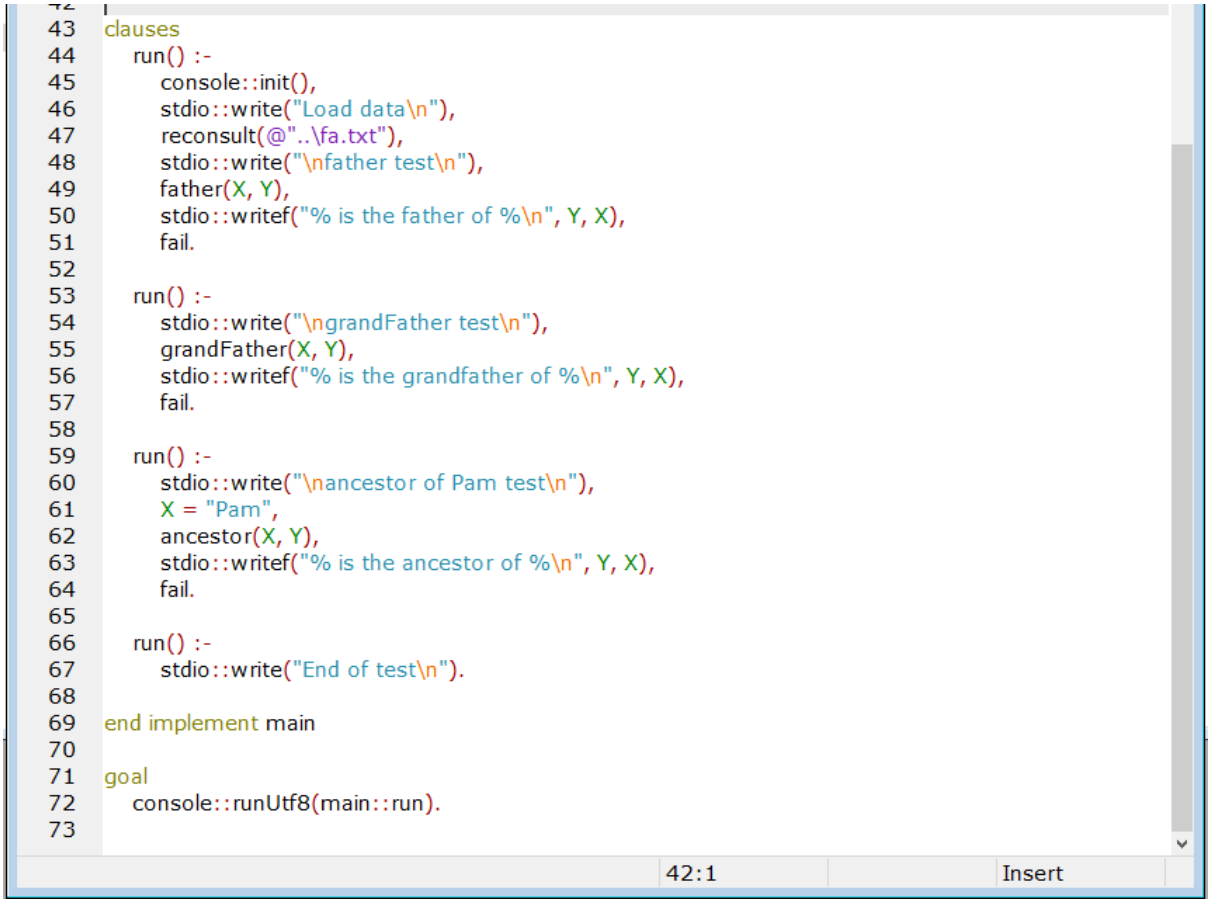
Συνολικά, η Visual Prolog είναι μια ισχυρή και ευέλικτη γλώσσα που είναι κατάλληλη για την ανάπτυξη έξυπνων εφαρμογών. Το σύστημα ανάπτυξης της περιλαμβάνει κανόνες, λογική και σχέσεις μεταξύ διαφορετικών αντικειμένων και δομών δεδομένων. Ωστόσο, μπορεί να απαιτήσει κάποια πρόσθετη εκμάθηση και προσπάθεια για προγραμματιστές που δεν είναι ήδη εξοικειωμένοι με την Prolog και τον λογικό προγραμματισμό. Αλλά μόλις εξοικειωθείτε με τη σύνταξη και τις δυνατότητές της, μπορείτε να δημιουργήσετε σύνθετες εφαρμογές γρήγορα και εύκολα.

Παρακάτω παρατίθεται πάραδειγμα υλοποίησης οικογενειακού δένδρου.

main.pro ()

```
% Copyright Tsarouchas Vasileios

implement main
    open core

domains
    gender = female; male.

class facts - familyDB
    person : (string Name, gender Gender).
    parent : (string Person, string Parent).

class predicates
    father : (string Person, string Father) nondeterm anyflow.
clauses
    father(Person, Father) :-
        parent(Person, Father),
        person(Father, male).

class predicates
    grandFather : (string Person [out], string GrandFather [out]) nondeterm.
clauses
    grandFather(Person, GrandFather) :-
        parent(Person, Parent),
        father(Parent, GrandFather).

class predicates
    ancestor : (string Person, string Ancestor [out]) nondeterm.
clauses
    ancestor(Person, Ancestor) :-
        parent(Person, Ancestor).
    ancestor(Person, Ancestor) :-
        parent(Person, P1),
        ancestor(P1, Ancestor).

class predicates
    reconsult : (string FileName).
clauses
    reconsult(FileName) :-
        retractFactDB(familyDB),
        file::consult(FileName, familyDB).
```

ΕΙΚΟΝΑ 1: ΠΑΡΑΔΕΙΓΜΑ ΟΙΚΟΓΕΝΕΙΑΚΟΥ ΔΕΝΤΡΟΥ ΣΤΗ VISUAL PROLOG



```
43  clauses
44      run() :-
45          console::init(),
46          stdio::write("Load data\n"),
47          reconsult(@"..\fa.txt"),
48          stdio::write("\nfather test\n"),
49          father(X, Y),
50          stdio::writef("% is the father of %\n", Y, X),
51          fail.
52
53      run() :-
54          stdio::write("\ngrandFather test\n"),
55          grandFather(X, Y),
56          stdio::writef("% is the grandfather of %\n", Y, X),
57          fail.
58
59      run() :-
60          stdio::write("\nancestor of Pam test\n"),
61          X = "Pam",
62          ancestor(X, Y),
63          stdio::writef("% is the ancestor of %\n", Y, X),
64          fail.
65
66      run() :-
67          stdio::write("End of test\n").
68
69  end implement main
70
71  goal
72      console::runUtf8(main::run).
73
                                    42:1                Insert
```

ΕΙΚΟΝΑ 2 : ΠΑΡΑΔΕΙΓΜΑ ΟΙΚΟΓΕΝΕΙΑΚΟΥ ΔΕΝΤΡΟΥ ΣΤΗ VISUAL PROLOG

Στην Visual Prolog, πρέπει να καθορίσετε τα πεδία και τα κατηγορήματα πριν από τον ορισμό των ρητρών. Πρέπει επίσης να προσθέσετε ένα τμήμα στόχου για την εκτέλεση ερωτημάτων κατά την εκτέλεση.

Το πρόγραμμα λειτουργεί όπως και η έκδοση Prolog, επιτρέποντάς σας να κάνετε ερωτήσεις σχετικά με τις σχέσεις μεταξύ των διαφόρων ατόμων στο οικογενειακό δέντρο. Η έξοδος θα εμφανίζεται στο παράθυρο της κονσόλας.

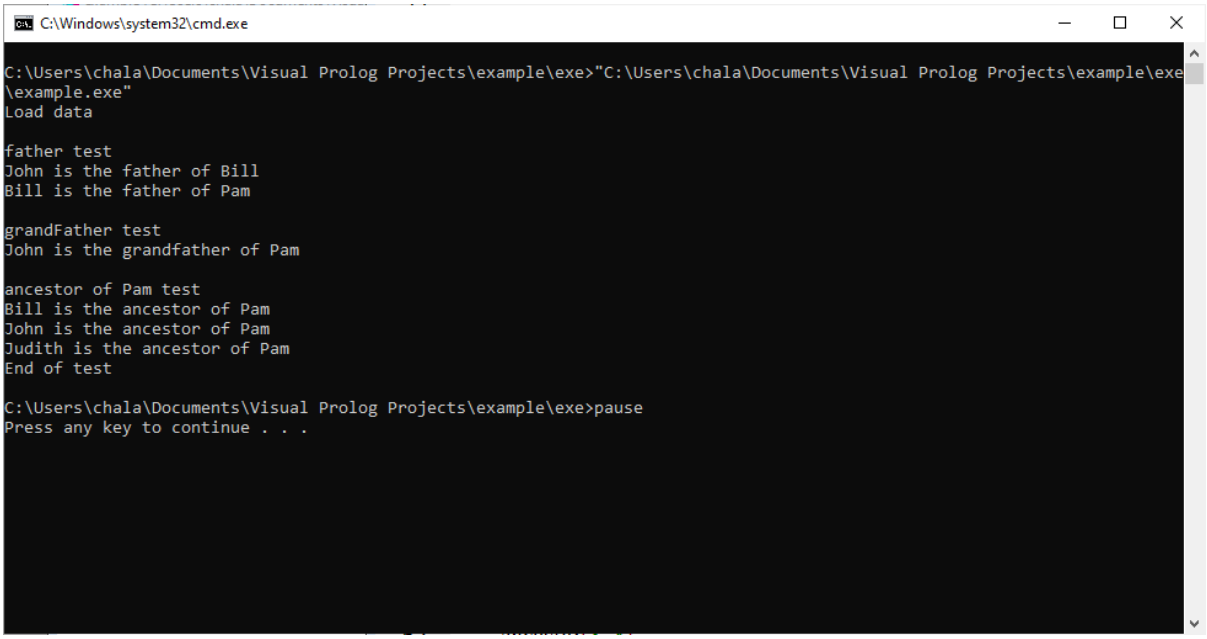
Ο κώδικας ξεκινά με τον ορισμό των πεδίων και των κατηγορημάτων. Στην Visual Prolog, ένας τομέας είναι ένας τρόπος ορισμού ενός τύπου ή ενός εύρους πιθανών τιμών για μια μεταβλητή. Σε αυτή την περίπτωση, ορίζουμε έναν τομέα που ονομάζεται "person" ως συμβολοσειρά για να αναπαραστήσουμε τα ονόματα των ατόμων στο οικογενειακό δέντρο. Ορίζουμε επίσης διάφορα κατηγορήματα για τους διάφορους τύπους σχέσεων και το φύλο.

Στη συνέχεια, ορίζουμε τις ρήτρες για το οικογενειακό δέντρο. Αυτό περιλαμβάνει τις σχέσεις γονέων, το φύλο κάθε προσώπου και τους κανόνες για τους διαφορετικούς τύπους σχέσεων. Οι κανόνες ορίζονται χρησιμοποιώντας τον τελεστή ":-", που σημαίνει "αν". Για

παράδειγμα, ο κανόνας του πατέρα λέει ότι αν ο X είναι γονέας του Y και ο X είναι άνδρας, τότε ο X είναι ο πατέρας του Y.

Τέλος, ορίζουμε ένα τμήμα στόχου για την εκτέλεση ερωτημάτων κατά την εκτέλεση. Σε αυτή την περίπτωση, έχουμε τρία παραδείγματα ερωτημάτων που θέτουν ερωτήσεις σχετικά με τις σχέσεις μεταξύ των ατόμων στο οικογενειακό δέντρο. Η συνάρτηση writeln χρησιμοποιείται για την εμφάνιση της εξόδου στο παράθυρο της κονσόλας.

Όταν εκτελείται το πρόγραμμα, η έξοδος θα εμφανιστεί στο παράθυρο της κονσόλας για κάθε ερώτημα.



ΕΙΚΟΝΑ 3 : ΑΠΟΤΕΛΕΣΜΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΟΙΚΟΓΕΝΕΙΑΚΟΥ ΔΕΝΤΡΟΥ

Για το παραπάνω αποτέλεσμα έπρεπε να δηλωθούν οι παρακάτω κανόνες, οι οποίοι είναι αποθηκευμένοι σε ένα αρχείο με κατάληξη .txt στον φάκελο του προγράμματος.

```
clauses
    person("Judith", female).
    person("Bill", male).
    person("John", male).
    person("Pam", female).
    parent("John", "Judith").
    parent("Bill", "John").
    parent("Pam", "Bill").
```

1.3.2 Java

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου. Αρχικά αναπτύχθηκε από τον James Gosling την δεκαετία του 1990, όταν ο ίδιος δούλευε για την εταιρεία Sun Microsystems. Το αρχικό της όνομα ήταν Oak. Το οποίο είχε πάρει από την βελανιδιά που είχε έξω από το γραφείο του ο James Gosling. Όμως, το όνομα (Oak) ήταν ήδη κατοχυρωμένο από άλλη επιχείρηση. Έτσι μετανομάστηκε σε Java, που ήταν και το όνομα του αγαπημένου καφέ των δημιουργών της. Για αυτό το λόγο το λογότυπο της εκτός από το όνομά της γλώσσας προγραμματισμούς έχει σχεδιασμένο και ένα φλυτζάνι ζεστού καφέ. Το 2010 όμως, η εταιρεία λογισμικού Oracle Corporation εξαγώρασε την εταιρεία Sun Microsystems μαζί με τις τεχνολογίες της. Οπότε από την δεκαετία του 2010 εώς και σήμερα η γλώσσα προγραμματισμού Java ανήκει στην εταιρεία λογισμικού Oracle Corporation. [11]

Ένα από τα βασικά χαρακτηριστικά της είναι η ανεξαρτησία της πλατφόρμας. Επειδή ο κώδικας της μεταγλωτίζεται σε bytecode που μπορεί να εκτελεστεί σε οποιαδήποτε πλατφόρμα, συμπεριλαμβανομένων των Windows, macOS, Linux και φορητών συσκευών, αρκεί να υπάρχει εγκατεστημένο το Java Virtual Machine (JVM). Άρα, ο κώδικας Java μπορεί να γραφεί μία φορά και να εκτελεστεί οπουδήποτε.

Είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, που σημαίνει ότι έχει σχεδιαστεί γύρω από την έννοια των αντικειμένων. Ένα αντικείμενο είναι μια αυτόνομη μονάδα κώδικα που περιέχει δεδομένα και συμπεριφορά και τα αντικείμενα μπορούν να αλληλεπιδράσουν μεταξύ τους για να εκτελέσουν εργασίες. Υποστηρίζει λειτουργίες όπως η κληρονομικότητα, ο πολυμορφισμός και η ενθυλάκωση, που μπορούν να βοηθήσουν τους προγραμματιστές να γράφουν καθαρό, επαναχρησιμοποιήσιμο, εύκολο στη διαχείριση και στη συντήρηση κώδικα.

Επιπλέον παρέχει εκτεταμένη υποστήριξη για multi-threading, η οποία επιτρέπει στους προγραμματιστές να γράφουν κώδικα που μπορεί να εκτελέσει πολλές εργασίες ταυτόχρονα. Αυτό διευκολύνει τη σύνταξη κώδικα ώστε να ανταποκρίνεται και να έχει καλή απόδοση, ιδιαίτερα σε εφαρμογές που απαιτούν πολύ υπολογισμό.

Η σύνταξη της είναι παρόμοια με αυτή της C και της C++, καθιστώντας εύκολη την εκμάθηση για προγραμματιστές με εμπειρία σε αυτές τις γλώσσες. Διαθέτει πολλά χαρακτηριστικά που διευκολύνουν τη σύνταξη και τη διατήρηση κώδικα, όπως η συλλογή

σκουπιδιών, ο ενσωματωμένος χειρισμός εξαιρέσεων και η εκτεταμένη υποστήριξη πολλαπλών νημάτων. Η Java χρησιμοποιεί έναν συλλέκτη απορριμμάτων για να διαχειρίζεται αυτόματα τη μνήμη, πράγμα που σημαίνει ότι οι προγραμματιστές δεν χρειάζεται να εκχωρούν και να εκχωρούν με μη αυτόματο τρόπο δεδομένα στη μνήμη. Αυτό καθιστά τον κώδικα Java πιο εύκολο στην εγγραφή και λιγότερο επιρρεπή σε σφάλματα που σχετίζονται με τη μνήμη.

Διαθέτει ένα μεγάλο και ενεργό οικοσύστημα εργαλείων, πλαισίων και βιβλιοθηκών. Μία μεγάλη τυπική βιβλιοθήκη κλάσεων και συναρτήσεων για την εκτέλεση κοινών εργασιών, όπως υποστήριξη για δικτύωση, είσοδο/έξοδο (I/O), επεξεργασία XML και άλλα. Αυτό μπορεί να εξοικονομήσει πολύ χρόνο και προσπάθεια στους προγραμματιστές, καθώς δεν χρειάζεται να γράψουν αυτές τις λειτουργίες από την αρχή.

Η Java χρησιμοποιείται ευρέως για την ανάπτυξη μιας ποικιλίας εφαρμογών, συμπεριλαμβανομένων εφαρμογών web, εφαρμογών για κινητά, εφαρμογών επιτραπέζιου υπολογιστή (Desktop Applications) και εταιρικού λογισμικού (Software Applications). Μερικά από τα πιο δημοφιλή πλαίσια (frameworks) και τεχνολογίες για την ανάπτυξη πολύπλοκων εφαρμογών Java περιλαμβάνουν τα Spring, Hibernate και Apache Struts, τα οποία παρέχουν ισχυρές δυνατότητες για ανάπτυξη ιστού, πρόσβαση σε δεδομένα και πολλά άλλα. Επιπλέον είναι ιδιαίτερα δημοφιλής στον κόσμο των επιχειρήσεων. Πολλές μεγάλες εταιρείες την χρησιμοποιούν για κρίσιμες εφαρμογές. Το λειτουργικό σύστημα Android είναι χτισμένο πάνω από μια τροποποιημένη έκδοση της, η οποία την έχει καταστήσει δημοφιλή επιλογή για την ανάπτυξη εφαρμογών για κινητά.

Ενώ η Java είναι μια δημοφιλής και ευέλικτη γλώσσα προγραμματισμού, έχει ορισμένα μειονεκτήματα. Μία από τις κύριες επικρίσεις της είναι ότι μπορεί να είναι πιο αργή από άλλες γλώσσες προγραμματισμού, ιδιαίτερα όταν πρόκειται για υπολογιστές υψηλής απόδοσης. Αυτό οφείλεται στην επιβάρυνση που συνεπάγεται η εκτέλεση κώδικα στο JVM και της αυτόματης διαχείρισης μνήμης που παρέχει η Java. Επειδή παρέχει αυτόματη διαχείριση μνήμης και άλλες αφαιρέσεις υψηλού επιπέδου, μερικές φορές μπορεί να είναι πιο δύσκολο να βελτιωθεί η απόδοση ή η χρήση μνήμης σε σύγκριση με γλώσσες χαμηλότερου επιπέδου όπως η C ή C++.

Αξίζει να σημειωθεί ότι είχε κάποια ζητήματα ασφαλείας στο παρελθόν, με ευπάθειες που ανακαλύφθηκαν στο Java Runtime Environment (JRE) που θα μπορούσαν να εκμεταλλευτούν οι εισβολείς. Ενώ η εταιρεία Oracle Corporation έχει αντιμετωπίσει πολλά από αυτά τα ζητήματα, η Java εξακολουθεί να θεωρείται πιθανός κίνδυνος ασφάλειας σε ορισμένα πλαίσια.

Επίσης η Java παρέχεται με την άδεια χρήσης Oracle Binary Code (BCL), η οποία έχει ορισμένους περιορισμούς σχετικά με τον τρόπο χρήσης και διανομής της Java. Αυτό μπορεί να είναι ανησυχητικό για ορισμένες επιχειρήσεις και προγραμματιστές που αναζητούν ένα πιο ανεκτικό μοντέλο αδειοδότησης.

Συνολικά, η Java είναι μια ισχυρή και ευέλικτη γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως σε μια ποικιλία βιομηχανιών και εφαρμογών. Η ευκολία χρήσης, η φορητότητα και το εκτεταμένο οικοσύστημα εργαλείων και πλαισίων το καθιστούν δημοφιλή επιλογή τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές. Όμως, έχει ορισμένα μειονεκτήματα που πρέπει να γνωρίζουν οι προγραμματιστές όταν επιλέγουν μια γλώσσα για ένα συγκεκριμένο έργο ή εφαρμογή. Για παράδειγμα, μπορεί να είναι αναλυτική και να απαιτεί πολύ κώδικα boilerplate, κάτι που μπορεί να κάνει πιο δύσκολη τη χρήση της από κάποιες άλλες γλώσσες προγραμματισμού. Ωστόσο, η Java παραμένει μια από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού στον κόσμο και η δημοτικότητά της δεν δείχνει σημάδια μείωσης.

Παρακάτω παρατίθεται ενα παράδειγματα υλοποίησης οικογενειακού δέντρου.

```java
import java.util.ArrayList;
import java.util.List;

public class Person {
    private String name;
    private boolean male;
    private List<Person> children = new ArrayList<>();

    public Person(String name, boolean male) {
        this.name = name;
        this.male = male;
    }

    public String getName() {
        return name;
    }

    public boolean isMale() {
        return male;
    }

    public void addChild(Person child) {
        children.add(child);
    }

    public List<Person> getChildren() {
        return children;
    }

    public boolean isParentOf(Person child) {
        return children.contains(child);
    }

    public boolean isGrandparentOf(Person grandchild) {
        for (Person child : children) {
            if (child.isParentOf(grandchild)) {
                return true;
            }
        }
        return false;
    }

    public static void main(String[] args) {
        // Define family tree
        Person john = new Person("John", true);
        Person mary = new Person("Mary", false);
        Person bob = new Person("Bob", true);
        Person susan = new Person("Susan", false);
        Person peter = new Person("Peter", true);

        john.addChild(mary);
        john.addChild(bob);
        mary.addChild(susan);
        bob.addChild(peter);

        // Query family tree
        System.out.println("John is the father of Mary: " + john.isParentOf(mary));
        System.out.println("John is the father of Bob: " + john.isParentOf(bob));
        System.out.println("Mary is the mother of Susan: " + mary.isParentOf(susan));
        System.out.println("Bob is the father of Peter: " + bob.isParentOf(peter));
        System.out.println("John is the grandfather of Susan: " + john.isGrandparentOf(susan));
        System.out.println("John is the grandfather of Peter: " + john.isGrandparentOf(peter));
    }
}
```

ΕΙΚΟΝΑ 4 : ΠΑΡΑΔΕΙΓΜΑ ΟΙΚΟΓΕΝΕΙΑΚΟΥ ΔΕΝΤΡΟΥ ΣΤΗ JAVA

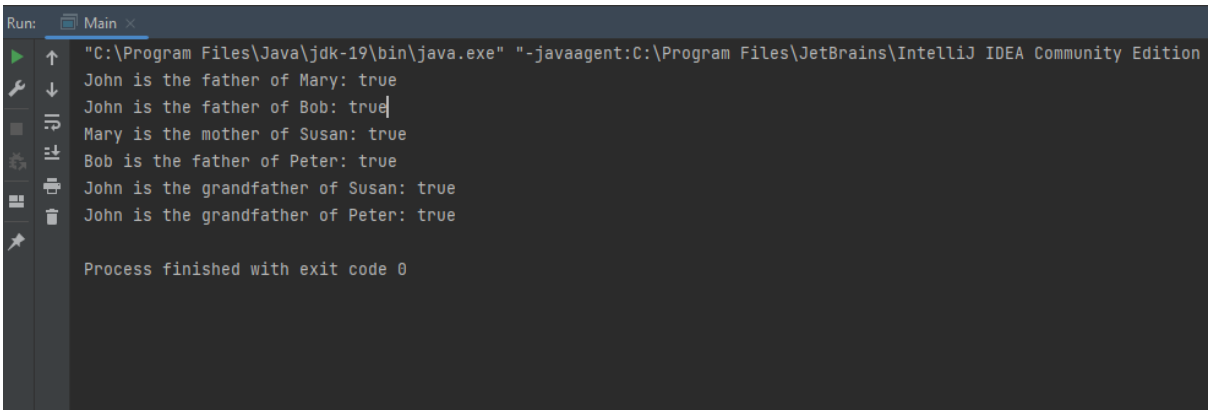
Σε αυτή την υλοποίηση, ορίζουμε μια κλάση Person που αντιπροσωπεύει ένα άτομο στο οικογενειακό δέντρο. Κάθε αντικείμενο Person έχει ένα όνομα, μια boolean μεταβλητή που

δείχνει αν το άτομο είναι αρσενικό ή θηλυκό, και μια λίστα με αντικείμενα Person που είναι τα παιδιά. Μπορούμε να προσθέσουμε αντικείμενα-παιδιά Person σε ένα αντικείμενο Person του γονέα χρησιμοποιώντας τη μέθοδο addChild().

Ορίζουμε επίσης δύο μεθόδους, τις isParentOf() και isGrandparentOf(), οι οποίες ελέγχουν αν το τρέχον αντικείμενο Person είναι γονέας ή παππούς ενός άλλου αντικειμένου Person. Αυτές οι μέθοδοι διατρέχουν αναδρομικά το οικογενειακό δέντρο ελέγχοντας τη λίστα παιδιών κάθε αντικειμένου Person.

Στη μέθοδο main(), δημιουργούμε αντικείμενα Person για κάθε άτομο στο οικογενειακό δέντρο και τα συνδέουμε χρησιμοποιώντας τη μέθοδο addChild(). Στη συνέχεια μπορούμε να αναζητήσουμε το οικογενειακό δέντρο καλώντας τις μεθόδους isParentOf() και isGrandparentOf() στα κατάλληλα αντικείμενα Person.

Μετά την εκτέλεση του παραπάνω κώδικα, προκύπτει το παρακάτω αποτέλεσμα.



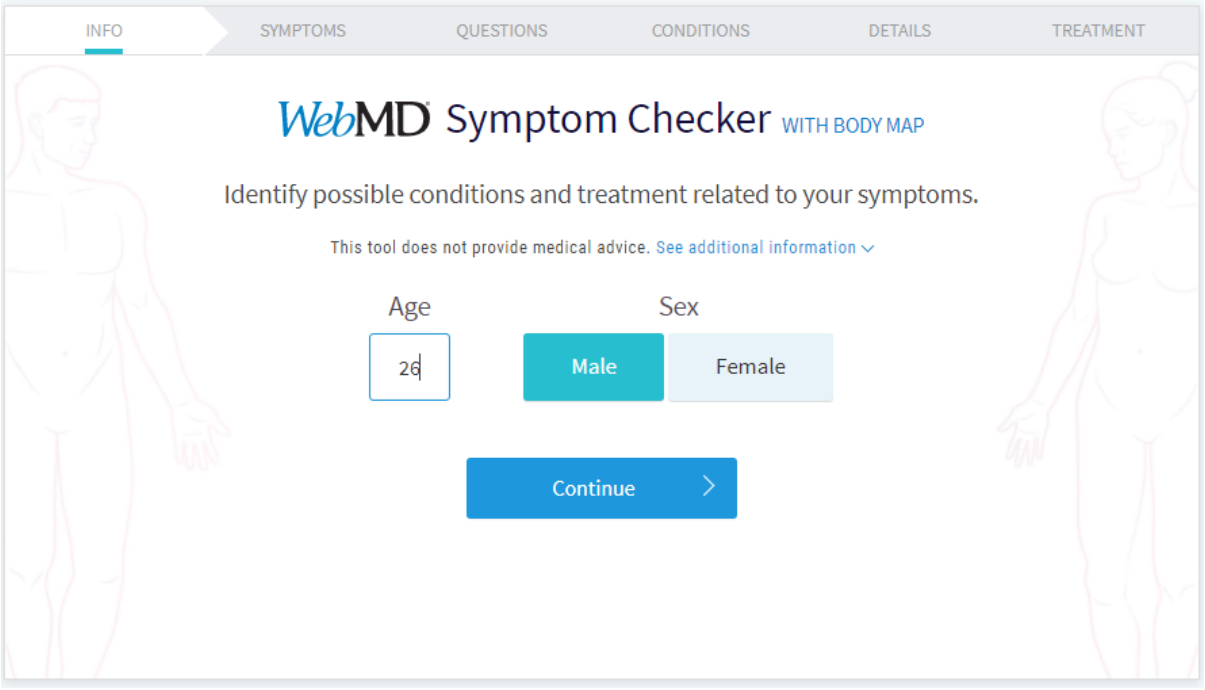
ΕΙΚΟΝΑ 5 : ΑΠΟΤΕΛΕΣΜΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΟΙΚΟΓΕΝΕΙΑΚΟΥ ΔΕΝΤΡΟΥ

1.3 Ερευνητικά δεδομένα

Στην παρούσα ενότητα θα αναφερθούν τρεις ιατρικές διαγνωστικές εφαρμογές. Οι οποίες έχουν σκοπό την διευκόλυνση του ασθενή για μία άμεση και πρώτη δίαγνωση.Οι δύο πρώτες εφαρμογές ζητούν από τον χρήστη-ασθενή να καταχωρήσει κάποια βασικά προσωπικά στοιχεία, καθώς και τα συμπτώματα που έχει. Όμως η τελευταία εφαρμογή που θα αναφερθεί έχει ως σκοπό πέρα από την βοήθεια που παρέχει στον ασθενεί, να μαθεί και να κατανοήσει από τη πάσχει. Να βοηθήσει και τους παρόχους υγειονομικής περίθαλψης να κάνουν ακριβή και σωστή διάγνωση για παθήσεις του δέρματος, των μαλλών και των νυχιών, χάρη στη μεγάλη βιβλιοθήκη εικόνων και βίντεο που κατέχει.

Ωστόσο, είναι σημαντικό να σημειωθεί ότι καμία από τις παρακάτω εφαρμογές δεν αντικαθιστούν τις ιατρικές συμβουλές από αδειοδοτημένο πάροχο υγειονομικής περίθαλψης και οι χρήστες θα πρέπει πάντα να συμβουλεύονται έναν επαγγελματία υγείας πριν λάβουν οποιαδήποτε ιατρική απόφαση.

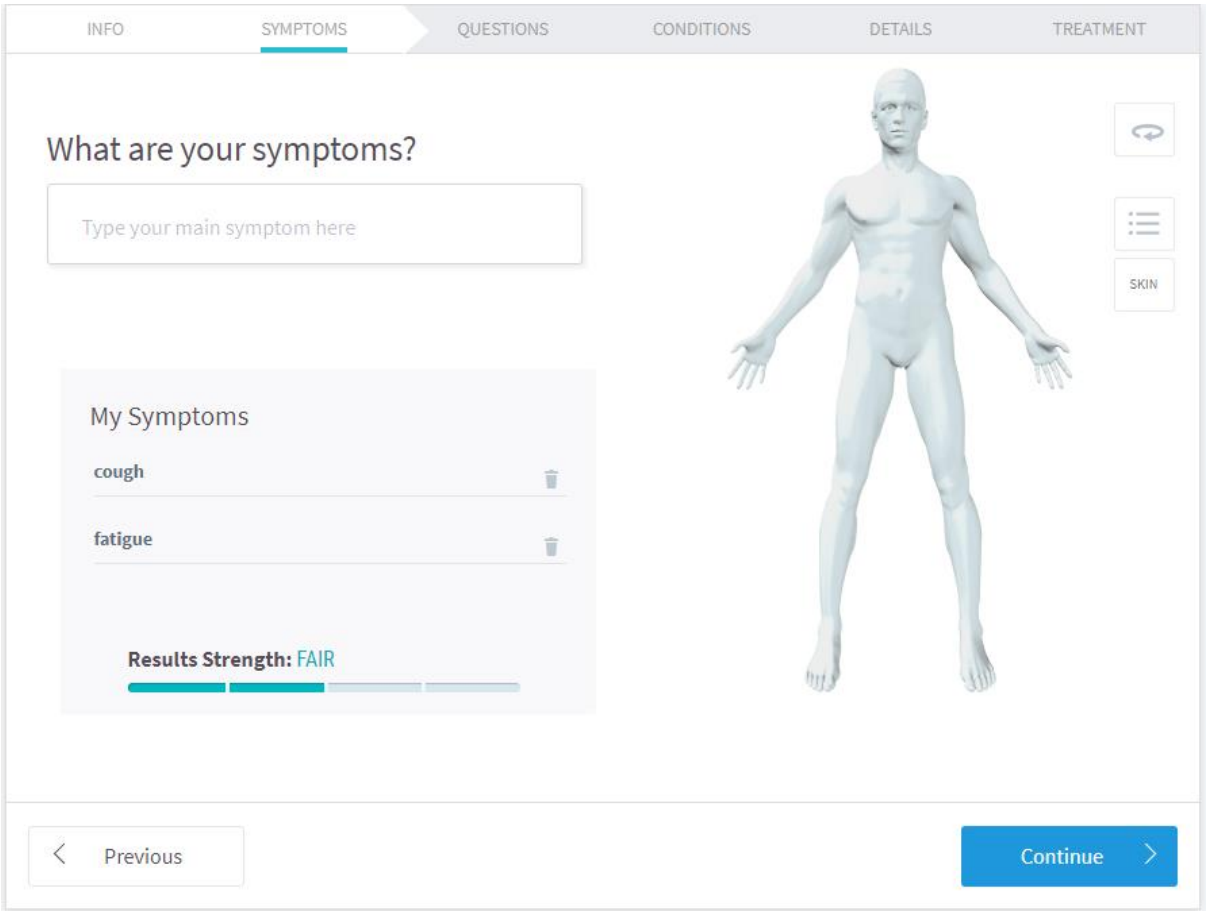
1.4.1 WebMD



ΕΙΚΟΝΑ 6 : ΕΦΑΡΜΟΓΗ WEBMD (ΠΗΓΗ: HTTPS://SYMPTOMS.WEBMD.COM/)

Το WebMD Symptom Checker είναι ένα διαδικτυακό εργαλείο που επιτρέπει στους χρήστες να εισάγουν πληροφορίες σχετικά με τα συμπτώματά τους για να λάβουν πιθανές διαγνώσεις και θεραπευτικές επιλογές. Πρόκειται για μια δωρεάν υπηρεσία που παρέχεται από το WebMD, έναν ιστότοπο πληροφοριών για την υγεία που παρέχει ιατρικές ειδήσεις, συμβουλές και εμπειρογνωμοσύνη. [12]

Για τη χρήση του Symptom Checker, οι χρήστες εισάγουν πληροφορίες σχετικά με τα συμπτώματά τους, συμπεριλαμβανομένης της πληγείσας περιοχής του σώματος, του τύπου του συμπτώματος και κάθε άλλης σχετικής πληροφορίας. Στη συνέχεια, το εργαλείο χρησιμοποιεί έναν αλγόριθμο για να αναλύσει τις πληροφορίες και να παράσχει έναν κατάλογο πιθανών καταστάσεων που μπορεί να προκαλούν τα συμπτώματα.

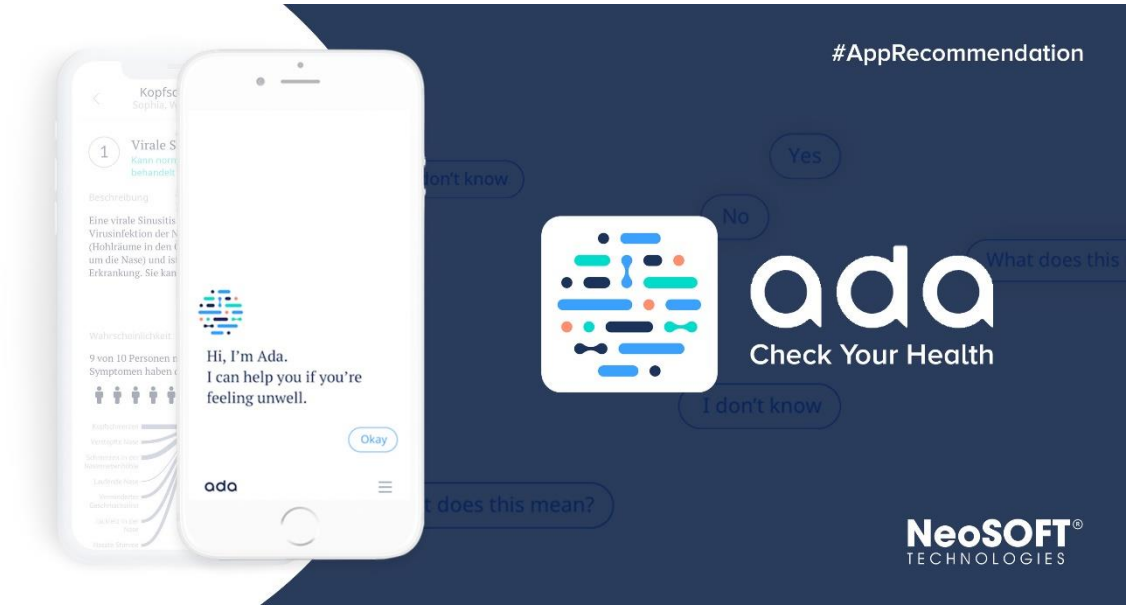


ΕΙΚΟΝΑ 7 : ΚΑΤΑΧΩΡΗΣΗ ΣΥΜΠΤΩΜΑΤΩΝ ΣΤΟ WEBMD (ΠΗΓΗ: HTTPS://SYMPTOMS.WEBMD.COM/)

Το Symptom Checker δεν έχει σκοπό να αντικαταστήσει τη συμβουλή ενός επαγγελματία ιατρού, αλλά να παρέχει στους χρήστες ένα σημείο εκκίνησης για την κατανόηση των συμπτωμάτων και των πιθανών αιτιών τους.

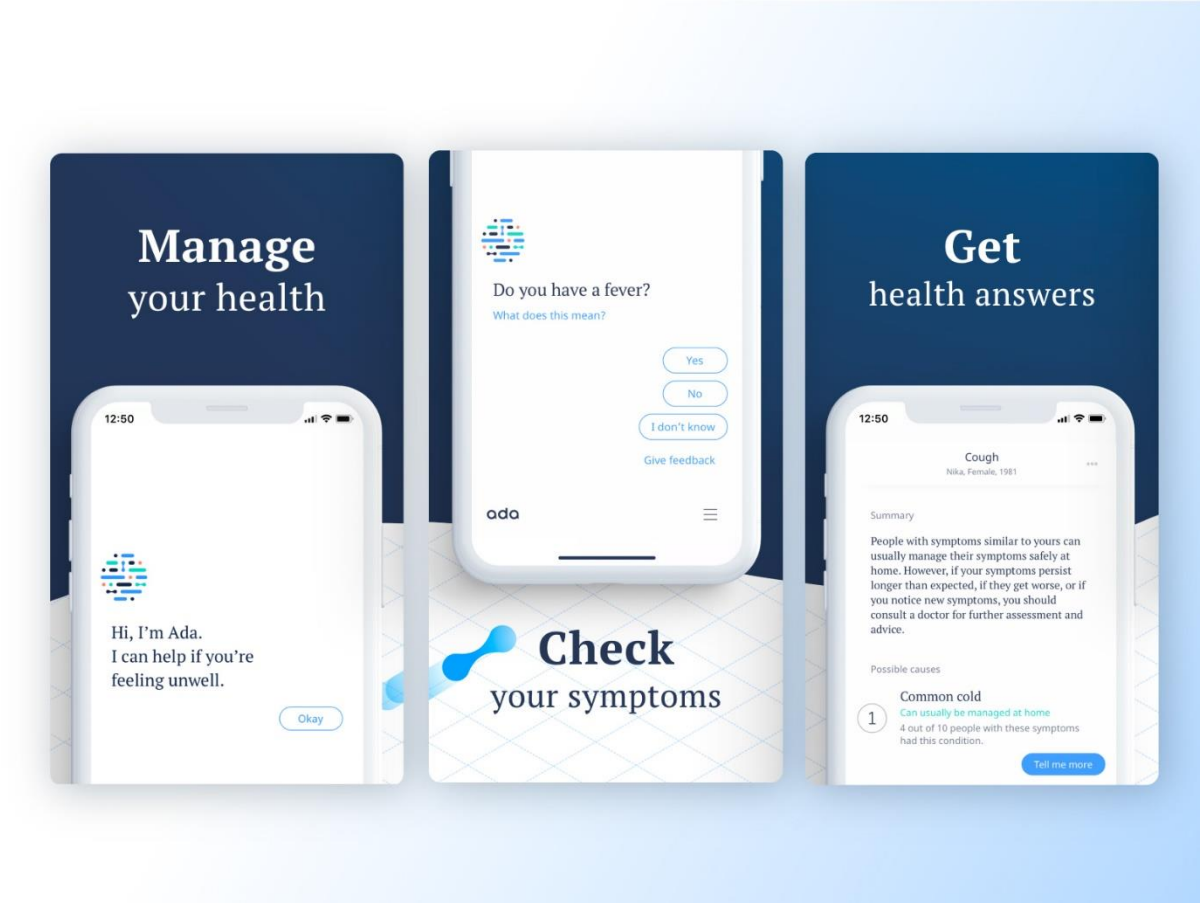
1.4.2 Ada

Το Ada είναι μια ψηφιακή πλατφόρμα υγείας που παρέχει στους χρήστες εξατομικευμένες αξιολογήσεις και πληροφορίες για την υγεία. Είναι ένας βοηθός υγείας με τεχνητή νοημοσύνη που βοηθά τους χρήστες να κατανοήσουν τα συμπτώματά τους και παρέχει συστάσεις για τη φροντίδα. [13]



ΕΙΚΟΝΑ 8 : ΕΦΑΡΜΟΓΗ ADA (ΠΗΓΗ: HTTPS://ADA.COM/)

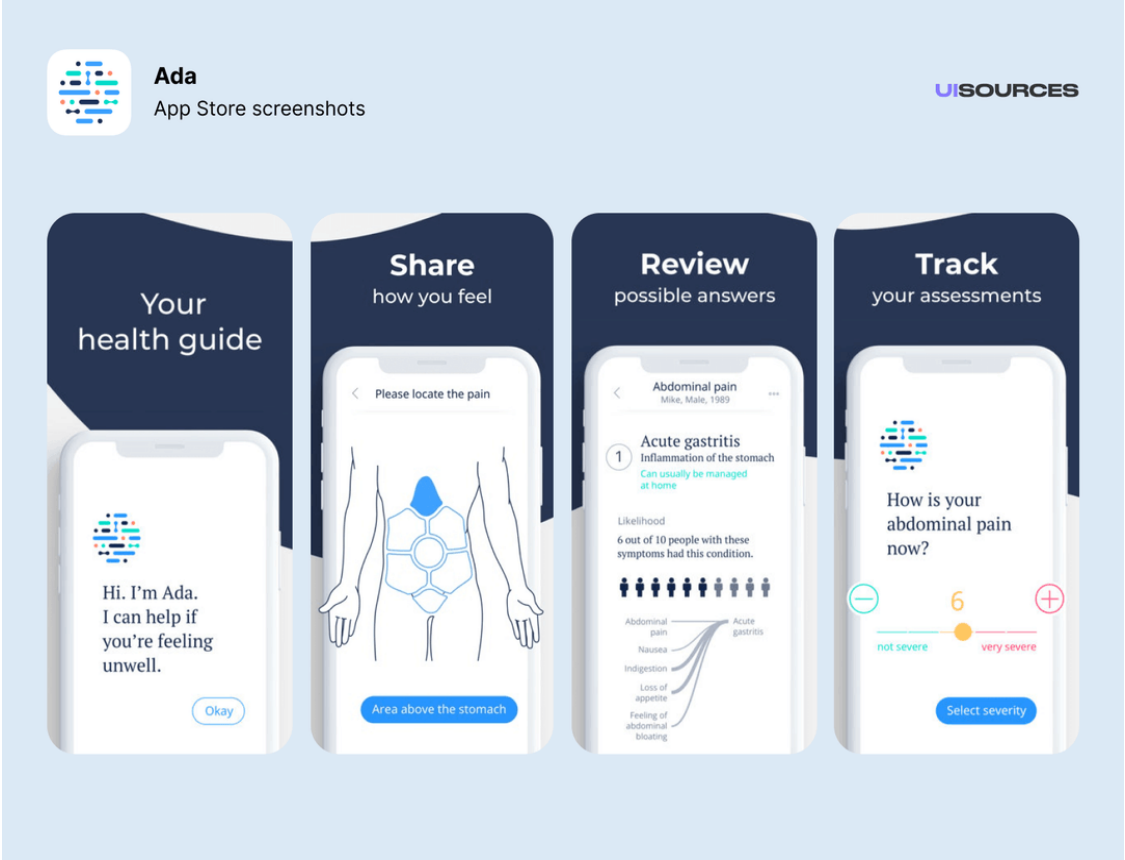
Για να χρησιμοποιήσουν το Ada, οι χρήστες δημιουργούν πρώτα έναν λογαριασμό και εισάγουν πληροφορίες σχετικά με το ιστορικό υγείας τους και τυχόν συμπτώματα που αντιμετωπίζουν. Η πλατφόρμα χρησιμοποιεί έναν ιδιόκτητο αλγόριθμο και επεξεργασία φυσικής γλώσσας για να αναλύσει τα συμπτώματα του χρήστη και να παρέχει μια εξατομικευμένη αξιολόγηση της υγείας του.



ΕΙΚΟΝΑ 9 : ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΗΣ ADA (ΠΗΓΗ: HTTPS://APPS.APPLE.COM/APP/ID1099986434?MT=8)

Οι αξιολογήσεις του Ada βασίζονται στην πιο πρόσφατη ιατρική έρευνα και ελέγχονται από μια ομάδα ιατρικών ειδικών για να διασφαλιστεί η ακρίβεια και η αξιοπιστία. Η πλατφόρμα παρέχει επίσης στους χρήστες πρόσβαση σε μια ολοκληρωμένη βιβλιοθήκη πληροφοριών και πόρων για την υγεία, συμπεριλαμβανομένων άρθρων, βίντεο και συμβουλών για τη διατήρηση της υγείας.

Ένα από τα βασικά χαρακτηριστικά του Ada είναι η ικανότητά του να συνδέει τους χρήστες με παρόχους υγειονομικής περίθαλψης, εάν είναι απαραίτητο. Με βάση την αξιολόγηση του χρήστη, η πλατφόρμα μπορεί να του συστήσει να αναζητήσει ιατρική φροντίδα και μπορεί να τον βοηθήσει να συνδεθεί με έναν τοπικό πάροχο υγειονομικής περίθαλψης. Το Ada μπορεί επίσης να παρέχει στους χρήστες εξατομικευμένες συστάσεις για αυτοφροντίδα, όπως οικιακές θεραπείες ή αλλαγές στον τρόπο ζωής.



ΕΙΚΟΝΑ 10 : ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ADA

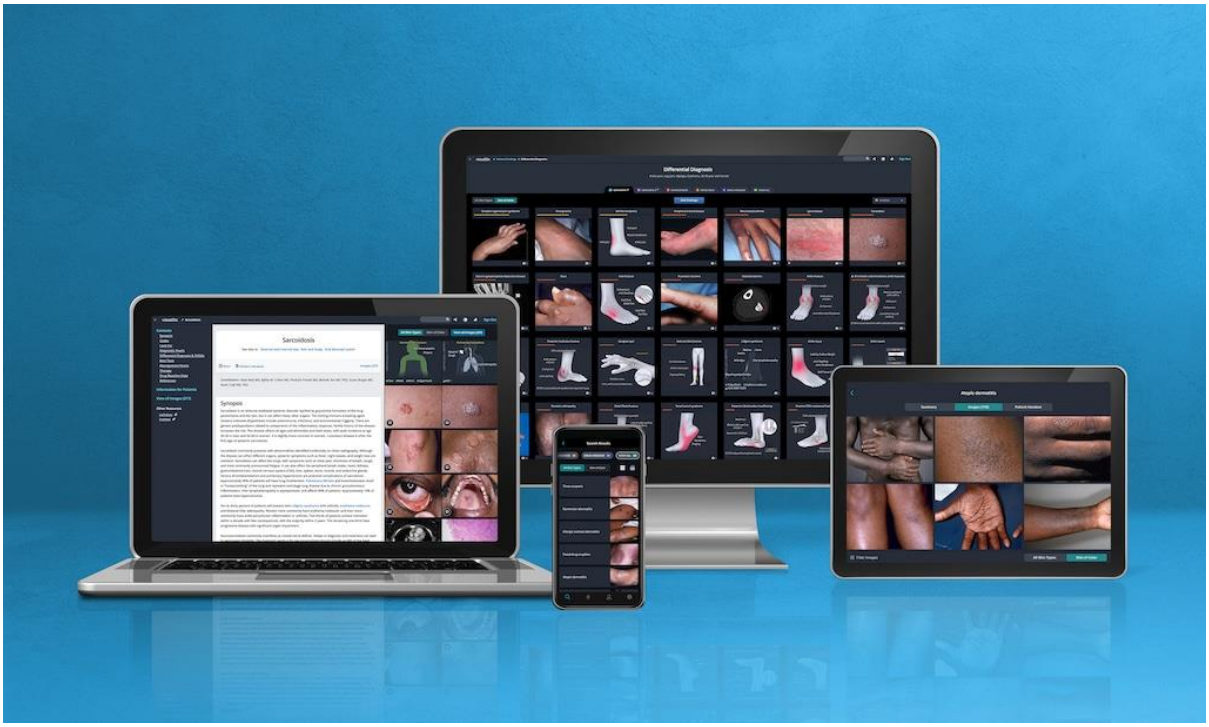
(ΠΗΓΗ: HTTPS://WWW.UISOURCES.COM/EXPLAINER/ADA-APP-STORE-SCREENSHOTS)

Συνολικά, το Ada είναι ένα πολύτιμο εργαλείο για τα άτομα που επιθυμούν να κατανοήσουν καλύτερα την υγεία τους και να λάβουν τεκμηριωμένες αποφάσεις σχετικά με τη φροντίδα τους. Παρέχοντας εξατομικευμένες αξιολογήσεις και συνδέοντας τους χρήστες με παρόχους υγειονομικής περίθαλψης, το Ada μπορεί να βοηθήσει τους χρήστες να πάρουν τον έλεγχο της υγείας και της ευεξίας τους.

1.4.3 VisualDx

ΕΙΚΟΝΑ 11 : ΕΦΑΡΜΟΓΗ VISUALDX (ΠΗΓΗ: HTTPS://WWW.VISUALDX.COM/)

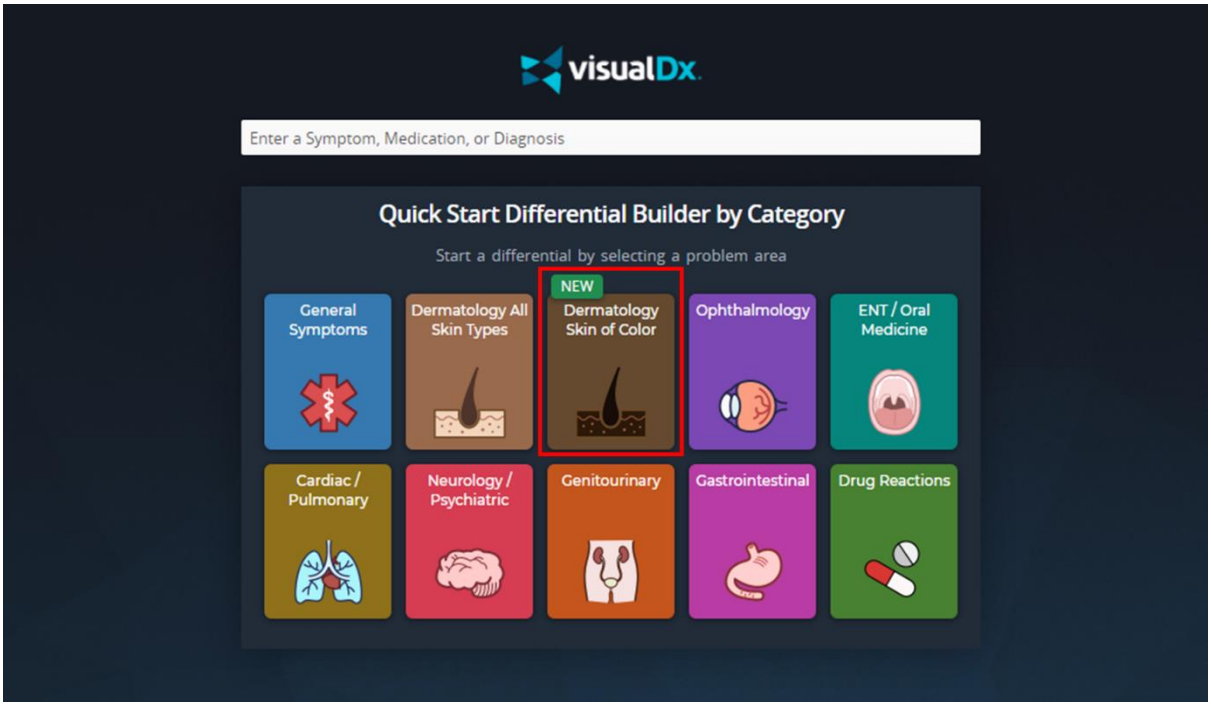
Το VisualDx είναι ένα σύστημα υποστήριξης κλινικών αποφάσεων που χρησιμοποιεί τεχνητή νοημοσύνη και μηχανική μάθηση για να βοηθήσει τους παρόχους υγειονομικής περίθαλψης να διαγνώσουν και να διαχειριστούν τις παθήσεις του δέρματος, των νυχιών και των μαλλιών. Πρόκειται για μια διαδικτυακή πλατφόρμα που χρησιμοποιείται σε περιβάλλοντα υγειονομικής περίθαλψης, όπως νοσοκομεία, κλινικές και τμήματα επειγόντων περιστατικών. [14]



ΕΙΚΟΝΑ 12 : ΔΥΝΑΤΟΤΗΤΕΣ ΤΗΣ VISUALDX (ΠΗΓΗ: HTTPS://WWW.UNMC.EDU/IEXCEL/VISUALIZATION/VISUAL-DX.HTML)

Ο ελεγκτής συμπτωμάτων του VisualDx επιτρέπει στους παρόχους υγειονομικής περίθαλψης να εισάγουν πληροφορίες σχετικά με τα συμπτώματα ενός ασθενούς, όπως η προσβεβλημένη περιοχή του σώματος, ο τύπος του συμπτώματος και οποιαδήποτε άλλη σχετική πληροφορία. Στη συνέχεια, η πλατφόρμα χρησιμοποιεί έναν αλγόριθμο για να αναλύσει τις πληροφορίες και να παράσχει έναν κατάλογο πιθανών διαγνώσεων.

Ένα από τα βασικά χαρακτηριστικά του VisualDx είναι η χρήση οπτικών βοηθημάτων για να βοηθήσει στη διάγνωση. Η πλατφόρμα περιλαμβάνει μια ολοκληρωμένη βιβλιοθήκη εικόνων και βίντεο που οι πάροχοι υγειονομικής περίθαλψης μπορούν να χρησιμοποιήσουν για να τα συγκρίνουν με τα συμπτώματα ενός ασθενούς και να κάνουν μια πιο ακριβή διάγνωση.



ΕΙΚΟΝΑ 13 : ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ VISUALDX (ΠΗΓΗ : HTTPS://WWW.UNMC.EDU/IEXCEL/VISUALIZATION/VISUAL-DX.HTML)

Το VisualDx περιλαμβάνει επίσης πληροφορίες σχετικά με τις θεραπευτικές επιλογές για κάθε πιθανή διάγνωση, καθώς και εκπαιδευτικό υλικό για τους ασθενείς, ώστε να κατανοήσουν καλύτερα την κατάστασή τους και τον τρόπο διαχείρισής της.

Συνολικά, το VisualDx είναι ένα πολύτιμο εργαλείο για τους παρόχους υγειονομικής περίθαλψης που επιθυμούν να κάνουν ακριβέστερες διαγνώσεις και να παρέχουν καλύτερη φροντίδα στους ασθενείς τους με παθήσεις του δέρματος, των νυχιών και των μαλλιών. Με τη χρήση μηχανικής μάθησης και οπτικών βοηθημάτων, το VisualDx μπορεί να βοηθήσει τους παρόχους να εντοπίσουν καταστάσεις που μπορεί να είναι δύσκολο να διαγνωστούν με παραδοσιακές μεθόδους και να παρέχουν πιο εξατομικευμένη και αποτελεσματική φροντίδα.

Κεφάλαιο 2ο

Στο παρόν κεφάλαιο θα γίνει επεξήγηση των εργαλείων σχεδιασμού, τα οποία είναι συνήθως απαραίτητα για έναν προγραμματιστή, ώστε να σχεδιάσει και να υλοποιήση μία εφαρμογή λογισμικού. Έπειτα θα παρουσιαστούν τα προγράμματα που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής μαζί με τις λέξεις – κλειδία. Τοσο για την γλώσσα προγραμματισμού Visual Prolog, όσο και για την Java. . Επιπλέον παρουσιάζεται και η βάση δεδομένων SQLite που ενσωματώθηκε στην Java. Κατά το κλείσιμο του κεφαλαίου γίνεται ανάλυση των διαγραμμάτων UML και ER.

2.1 Εργαλεία Σχεδιασμού

Για να σχεδιαστεί και να υλοποιηθεί μια εφαρμογή λογισμικού, ένας προγραμματιστής χρειάζεται συνήθως τα ακόλουθα εργαλεία :

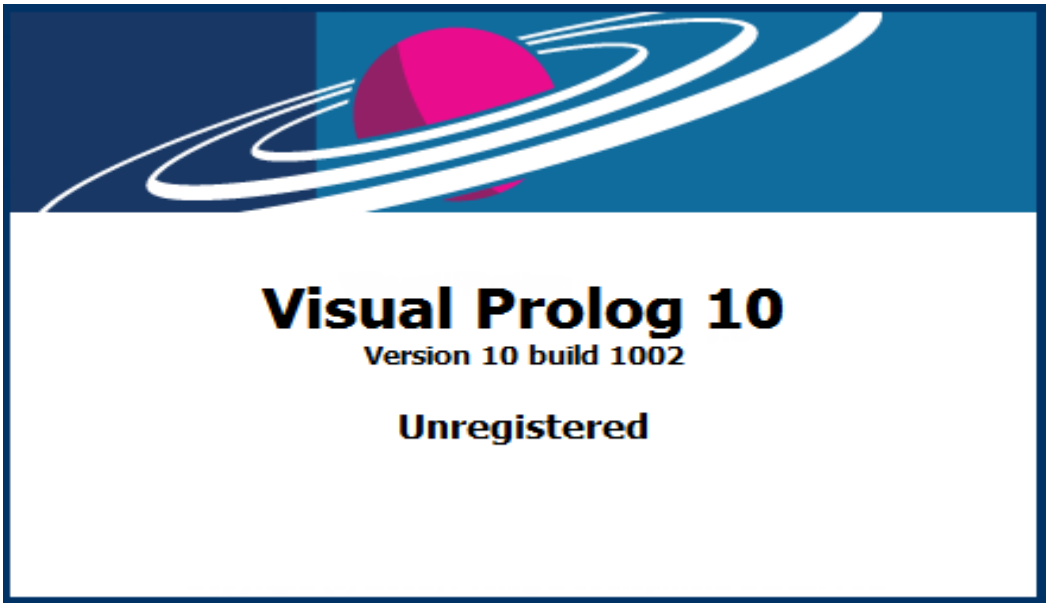
- Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE): Πρόκειται για μια εφαρμογή λογισμικού που παρέχει ένα σύνολο εργαλείων που βοηθούν τους προγραμματιστές να γράφουν, να δοκιμάζουν και να αποσφαλματώνουν τον κώδικά τους. Τα IDE συνήθως περιλαμβάνουν χαρακτηριστικά όπως επεξεργαστές κώδικα, μεταγλωττιστές, προγράμματα εντοπισμού σφαλμάτων και συστήματα ελέγχου εκδόσεων.
- Γλώσσες προγραμματισμού: Οι προγραμματιστές πρέπει να επιλέξουν μια γλώσσα προγραμματισμού που ταιριάζει καλύτερα στις απαιτήσεις της εφαρμογής τους. Ορισμένες δημοφιλείς γλώσσες προγραμματισμού είναι η Python, η Java, η C++ και η JavaScript.
- Πλαίσια και βιβλιοθήκες: Πρόκειται για προκατασκευασμένα κομμάτια κώδικα που οι προγραμματιστές μπορούν να χρησιμοποιήσουν για να επιταχύνουν την ανάπτυξη και να απλοποιήσουν πολύπλοκες εργασίες. Τα πλαίσια και οι βιβλιοθήκες μπορούν να χρησιμοποιηθούν για διάφορους σκοπούς, όπως η ανάπτυξη ιστοσελίδων, η ανάπτυξη εφαρμογών για κινητά και η ανάπτυξη παιχνιδιών.
- Βάσεις δεδομένων: Οι εφαρμογές πρέπει συχνά να αποθηκεύουν δεδομένα και οι προγραμματιστές πρέπει να επιλέξουν ένα σύστημα διαχείρισης βάσεων δεδομένων που ταιριάζει καλύτερα στις απαιτήσεις της εφαρμογής τους. Ορισμένες

δημοφιλείς βάσεις δεδομένων είναι η MySQL, SQLite, η PostgreSQL και η MongoDB.

- Εργαλεία δοκιμών: Είναι σημαντικό να δοκιμάζεται διεξοδικά μια εφαρμογή για να διασφαλιστεί ότι λειτουργεί όπως αναμένεται. Οι προγραμματιστές χρησιμοποιούν εργαλεία δοκιμών για να αυτοματοποιήσουν τη διαδικασία δοκιμών, να εντοπίσουν και να διορθώσουν σφάλματα.
- Εργαλεία συνεργασίας και επικοινωνίας: Οι προγραμματιστές εργάζονται συχνά σε ομάδες και χρειάζονται εργαλεία για να συνεργάζονται και να επικοινωνούν αποτελεσματικά. Ορισμένα δημοφιλή εργαλεία συνεργασίας και επικοινωνίας περιλαμβάνουν το Azure DevOps Server, το Bitbucket, το GitLab και το GitHub.

2.1.1 Visual Prolog 10 personal edition

Το Visual Prolog 10 Personal Edition είναι μια δωρεάν έκδοση για το περιβάλλον ανάπτυξης της γλώσσας προγραμματισμού Visual Prolog, η οποία είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που βασίζεται στον λογικό προγραμματισμό. Η έκδοση για προσωπική χρήση είναι μια ελαφριά έκδοση πλήρους λογισμικού, το οποίο διατίθεται προς αγορά. Πρόκειται για μια απλοποιημένη έκδοση που είναι κατάλληλη για την εκμάθηση και την εξάσκηση των βασικών αρχών του προγραμματισμού Visual Prolog.



ΕΙΚΟΝΑ 14 : VISUAL PROLOG

Είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για τον προγραμματισμό. Παρέχει ένα πλήρες σύνολο εργαλείων για την ανάπτυξη και την αποσφαλμάτωση προγραμμάτων, συμπεριλαμβανομένου ενός επεξεργαστή, ενός μεταγλωττιστή, ενός αποσφαλματωτή και ενός συστήματος διαχείρισης έργου.

Έχει σχεδιαστεί για να διευκολύνει τους προγραμματιστές να γράφουν και να δοκιμάζουν τον κώδικα τους. Ο επεξεργαστής (editor) του κώδικα παρέχει υπογράμμιση συντακτικού, αναδίπλωση κώδικα και αυτόματη συμπλήρωση, η οποία βοηθά τους προγραμματιστές να γράφουν κώδικα γρήγορα και αποτελεσματικά. Το IDE περιλαμβάνει επίσης έναν ενσωματωμένο αποσφαλματωτή (debugger), ο οποίος επιτρέπει στους προγραμματιστές να εξετάζουν βήμα προς βήμα τον κώδικά τους για να εντοπίζουν και να διορθώνουν σφάλματα.

Επίσης παρέχει μια σειρά από βιβλιοθήκες και εργαλεία που μπορούν να χρησιμοποιηθούν για την ανάπτυξη μιας ποικιλίας εφαρμογών, συμπεριλαμβανομένων συστημάτων τεχνητής νοημοσύνης, έμπειρων συστημάτων και εφαρμογών επεξεργασίας φυσικής γλώσσας.

Ένα από τα βασικά χαρακτηριστικά της Visual Prolog είναι η ικανότητά της να υποστηρίζει έννοιες αντικειμενοστραφούς προγραμματισμού (OOP). Επιτρέπει στους προγραμματιστές να δημιουργούν κλάσεις και αντικείμενα, τα οποία μπορούν να χρησιμοποιηθούν για την οργάνωση του κώδικα και τη δημιουργία σύνθετων δομών δεδομένων. Αυτό διευκολύνει την ανάπτυξη μεγάλων και πολύπλοκων εφαρμογών.

Κατά τη δημιουργία μιας εφαρμογής με το Visual Prolog, η δομή της εφαρμογής ακολουθεί συνήθως το παράδειγμα του λογικού προγραμματισμού, το οποίο βασίζεται στον ορισμό σχέσεων μεταξύ οντοτήτων δεδομένων.

Γενικά, μια εφαρμογή αποτελείται από μια συλλογή γεγονότων και κανόνων που καθορίζουν τη γνώση και τη συμπεριφορά της εφαρμογής. Τα γεγονότα αντιπροσωπεύουν τις οντότητες δεδομένων και τις σχέσεις τους, ενώ οι κανόνες καθορίζουν τον τρόπο με τον οποίο η εφαρμογή επεξεργάζεται τα δεδομένα και εκτελεί διάφορες λειτουργίες.

Στην Visual Prolog, η δομή της εφαρμογής οργανώνεται συνήθως σε ενότητες, οι οποίες είναι συλλογές γεγονότων και κανόνων που αφορούν μια συγκεκριμένη πτυχή της εφαρμογής. Οι ενότητες μπορούν να οργανωθούν ιεραρχικά, με υποενότητες που αντιπροσωπεύουν πιο συγκεκριμένες πτυχές της εφαρμογής.

Εκτός από τις ενότητες, οι εφαρμογές Visual Prolog μπορούν επίσης να περιλαμβάνουν εξωτερικές βιβλιοθήκες και στοιχεία που παρέχουν πρόσθετη λειτουργικότητα. Αυτές οι βιβλιοθήκες μπορούν να φορτωθούν στην εφαρμογή και να χρησιμοποιηθούν παράλληλα με τις δικές της ενότητες.

Συνολικά, η Visual Prolog είναι μια ισχυρή και ευέλικτη γλώσσα προγραμματισμού που είναι κατάλληλη για την ανάπτυξη εφαρμογών τεχνητής νοημοσύνης και έμπειρων συστημάτων. Αποτελεί ιδανική επιλογή για άτομα που ενδιαφέρονται να μάθουν περισσότερα για τον λογικό προγραμματισμό και τα έμπειρα συστήματα, καθώς και για προγραμματιστές που επιθυμούν να δημιουργήσουν εφαρμογές τεχνητής νοημοσύνης. Η δομή μιας εφαρμογής που δημιουργείται με το Visual Prolog εξαρτάται από τις συγκεκριμένες απαιτήσεις της εφαρμογής και τις σχεδιαστικές επιλογές που κάνει ο προγραμματιστής. Ωστόσο, γενικά θα ακολουθεί το παράδειγμα του λογικού προγραμματισμού και θα είναι οργανωμένη σε ενότητες που καθορίζουν τη γνώση και τη συμπεριφορά της εφαρμογής. [15]

Στη συνέχεια αναφέρονται οι λέξεις – κλειδιά για το περιβάλλον ανάπτυξης της Visual Prolog.

domains : Χρησιμοποιείται για την επισήμανση ενός τμήματος που δηλώνει τους τομείς που θα χρησιμοποιηθούν στον κώδικα. Ένας τομέας περιγράφει π.χ. ένα εύρος τιμών για μεταβλητές ή ένα είδος κατηγορημάτων. Γνωστοί τομείς είναι ο ακέραιος, ο πραγματικός και η συμβολοσειρά. Υπάρχουν πολλές παραλλαγές για τη σύνταξη τέτοιων δηλώσεων τομέων και καλύπτουν όλα τα πιθανά είδη τομέων.

facts : Δηλώνει ένα τμήμα, στο οποίο δηλώνονται τα γεγονότα που θα χρησιμοποιηθούν αργότερα στον κώδικα του προγράμματος. Ένα γεγονός μπορεί να είναι μια απλή μεταβλητή, αλλά μπορεί επίσης να είναι μία συνάρτηση που διαχειρίζεται πολλές τιμές ή και άλλες συναρτήσεις. Κάθε fact δηλώνεται με το όνομά του και στην περίπτωση ενός συναρτητή, τα ορίσματα που χρησιμοποιούνται για τα αντίστοιχα γεγονότα, μαζί με τα πεδία στα οποία ανήκουν αυτά τα ορίσματα.

class facts : Χρησιμοποιείται για τον ορισμό των class facts, τα οποία χρησιμοποιούνται για τον ορισμό των ιδιοτήτων και των συμπεριφορών των κλάσεων σε ένα πλαίσιο αντικειμενοστραφούς προγραμματισμού. Που επιτρέπει στους προγραμματιστές να ορίζουν κλάσεις και αντικείμενα με υψηλό βαθμό ευελιξίας και ελέγχου.

predicates : Είναι ένα χρήσιμο εργαλείο για την οργάνωση και τον ορισμό συνόλων σχετικών predicates μέσα σε μια ενότητα της Visual Prolog. Με τη χρήση των predicates, οι προγραμματιστές μπορούν να δημιουργήσουν ευέλικτα και ισχυρά προγράμματα που είναι κατάλληλα για ένα ευρύ φάσμα εφαρμογών.

clauses : Επιτρέπει στους προγραμματιστές να ορίζουν τους κανόνες και τα γεγονότα που συνθέτουν τα προγράμματά τους. Με την ανάλυση πολύπλοκων κατηγορημάτων σε απλούστερες ρήτρες, οι προγραμματιστές μπορούν να δημιουργήσουν πιο ισχυρές και ευέλικτες εφαρμογές που είναι κατάλληλες για ένα ευρύ φάσμα περιπτώσεων χρήσης.

class : Χρησιμοποιείται για τον ορισμό μιας νέας κλάσης, η οποία είναι ένα σχέδιο ή πρότυπο για τη δημιουργία αντικειμένων. Η λέξη-κλειδί class ακολουθείται από το όνομα της κλάσης και το σώμα της κλάσης με την αρχικοποίηση της (implement) και την οριστικοποίηση της (end implement).

implement : Χρησιμοποιείται για την έναρξη του τμήματος υλοποίησης μιας κλάσης. Μετά τη λέξη-κλειδί "implement", οι μέθοδοι και τα χαρακτηριστικά της κλάσης μπορούν να οριστούν και να υλοποιηθούν.

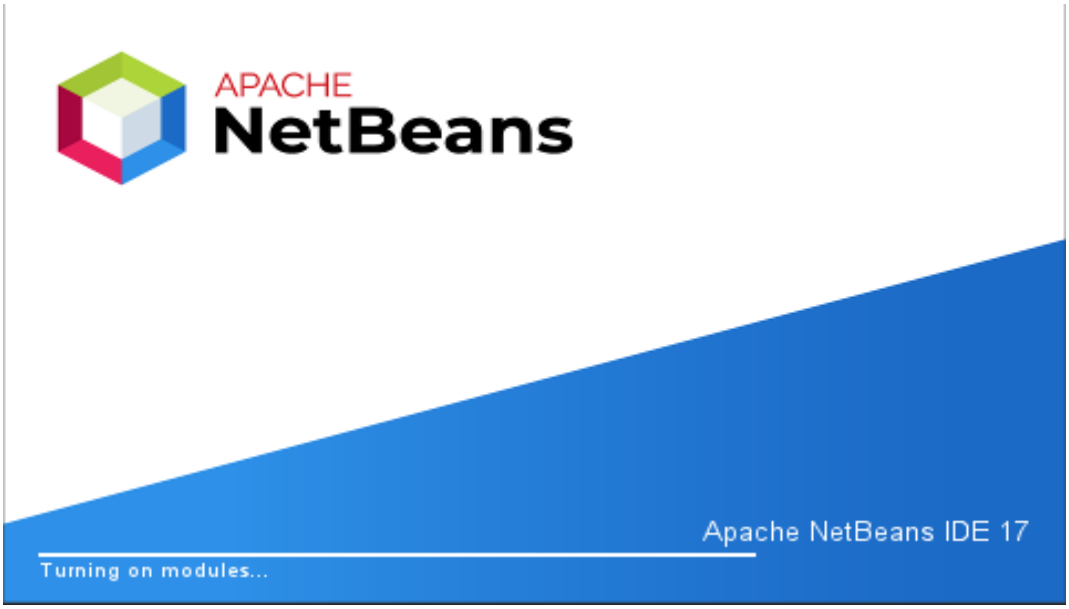
end implement : Χρησιμοποιείται για το τέλος του τμήματος υλοποίησης της κλάσης. Αφού οριστεί και υλοποιηθεί το τμήμα υλοποίησης, η κλάση μπορεί να χρησιμοποιηθεί για τη δημιουργία αντικειμένων και την εκτέλεση λειτουργιών με βάση τις μεθόδους και τα χαρακτηριστικά που έχουν οριστεί.

constants : Χρησιμοποιείται για τον ορισμό σταθερών τιμών που δεν μπορούν να τροποποιηθούν ή να αλλάξουν κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Οι σταθερές είναι παρόμοιες με τις μεταβλητές στο ότι κρατούν τιμές, αλλά σε αντίθεση με τις μεταβλητές, οι τιμές τους δεν μπορούν να τροποποιηθούν αφού οριστούν. Για παράδειγμα, μπορούν να βοηθήσουν να γίνει ο κώδικας πιο ευανάγνωστος παρέχοντας ουσιαστικά ονόματα για τιμές που διαφορετικά θα ήταν δύσκολο να κατανοηθούν. Μπορούν επίσης να βοηθήσουν να διασφαλιστεί ότι ορισμένες τιμές είναι συνεπείς σε ολόκληρο το πρόγραμμα, γεγονός που μπορεί να μειώσει την πιθανότητα σφαλμάτων.

open : Χρησιμοποιείται για την εισαγωγή ενοτήτων ή χώρων ονομάτων στο τρέχον πρόγραμμα ή στο αρχείο πηγής. Όταν χρησιμοποιείτε τη λέξη-κλειδί "open", ουσιαστικά λέτε στον μεταγλωττιστή να καταστήσει διαθέσιμα όλα τα αντικείμενα που ορίζονται στην εισαγόμενη ενότητα στο πρόγραμμα ή στο αρχείο του πηγαίου κώδικα.

2.2.2 Apache NetBeans 17

Το Apache NetBeans 17 είναι ένα περιβάλλον ολοκληρωμένης ανάπτυξης (IDE) ανοικτού κώδικα που χρησιμοποιείται για την ανάπτυξη εφαρμογών λογισμικού σε διάφορες γλώσσες προγραμματισμού. Συντηρείται από το Ίδρυμα Λογισμικού Apache και διατίθεται υπό την Άδεια Apache License 2.0.



ΕΙΚΟΝΑ 15 : APACHE NETBEANS

Το Apache NetBeans παρέχει ένα πλούσιο σύνολο εργαλείων και χαρακτηριστικών που διευκολύνουν τους προγραμματιστές να γράφουν, να δοκιμάζουν και να αναπτύσσουν εφαρμογές. Υποστηρίζει διάφορες γλώσσες προγραμματισμού, όπως Java, C, C++, PHP και HTML, μεταξύ άλλων. Το IDE παρέχει έναν επεξεργαστή κώδικα με δυνατότητες επισήμανσης συντακτικού, αυτόματης συμπλήρωσης και αναδιαμόρφωσης κώδικα, γεγονός που διευκολύνει τους προγραμματιστές να γράφουν κώδικα γρήγορα και αποτελεσματικά.

Ένα από τα βασικά χαρακτηριστικά του Apache NetBeans είναι η υποστήριξή του για πολλαπλές πλατφόρμες, συμπεριλαμβανομένων των Windows, macOS και Linux. Αυτό σημαίνει ότι οι προγραμματιστές μπορούν να χρησιμοποιούν το IDE στο λειτουργικό σύστημα της προτίμησής τους χωρίς προβλήματα.

Το Apache NetBeans παρέχει επίσης μια σειρά από plugins που μπορούν να χρησιμοποιηθούν για την επέκταση της λειτουργικότητάς του. Αυτά τα πρόσθετα

περιλαμβάνουν, μεταξύ άλλων, υποστήριξη για πρόσθετες γλώσσες προγραμματισμού, συστήματα ελέγχου εκδόσεων και διακομιστές εφαρμογών.

Το IDE περιλαμβάνει επίσης έναν ενσωματωμένο αποσφαλματωτή, ο οποίος επιτρέπει στους προγραμματιστές να εντοπίζουν και να διορθώνουν γρήγορα τα σφάλματα στον κώδικά τους. Επιπλέον, παρέχει εργαλεία για τη δημιουργία προφίλ και τον έλεγχο εφαρμογών, τα οποία μπορούν να βοηθήσουν τους προγραμματιστές να διασφαλίσουν ότι οι εφαρμογές τους είναι βελτιστοποιημένες και έχουν καλή απόδοση.

Συνολικά, το Apache NetBeans είναι ένα ισχυρό και ευέλικτο IDE που είναι κατάλληλο για ένα ευρύ φάσμα έργων ανάπτυξης. Είναι δημοφιλές μεταξύ των προγραμματιστών για την ευκολία χρήσης, την επεκτασιμότητα και την υποστήριξη πολλαπλών πλατφορμών. [16]

Κατά τη δημιουργία μιας εφαρμογής JavaFX με το NetBeans, η δομή της εφαρμογής θα εξαρτηθεί από το συγκεκριμένο πρότυπο έργου που θα επιλεξεί ο εκάστοτε προγραμματιστής. Ωστόσο, σε γενικές γραμμές, μια εφαρμογή JavaFX ακολουθεί το πρότυπο αρχιτεκτονικής Model-View-Controller (MVC). Συνολικά, η δομή μιας εφαρμογής θα είναι παρόμοια με αυτή άλλων εφαρμογών Java, αλλά θα περιλαμβάνει πρόσθετα στοιχεία ειδικά για το πλαίσιο JavaFX.

Η εφαρμογή που δημιουργήθηκε στα πλαίσια της εργασίας και στο γραφικό περιβάλλον που προαναφέρθηκε. Χρησιμοποιεί ακριβώς το ίδιο πρότυπο. Οι λέξεις – κλειδία θα αναλυθούν παρακάτω.

Model : Το Μοντέλο αναπαριστά τα δεδομένα και την επιχειρησιακή λογική της εφαρμογής και είναι υπεύθυνο για τη διαχείριση της κατάστασης της εφαρμογής. Συνήθως περιλαμβάνει κλάσεις (class) που καθορίζουν τις δομές δεδομένων και τους αλγορίθμους που χρησιμοποιούνται από την εφαρμογή.

View : Η προβολή αντιπροσωπεύει τη διεπαφή χρήστη της εφαρμογής και είναι υπεύθυνη για την εμφάνιση δεδομένων στον χρήστη και τη λήψη των εισροών του χρήστη. Σε μια εφαρμογή JavaFX, η προβολή συνήθως ορίζεται χρησιμοποιώντας την FXML, μια γλώσσα σήμανσης για την περιγραφή διεπαφών χρήστη.

Controller : Ο ελεγκτής ενεργεί ως ενδιάμεσος μεταξύ του μοντέλου και της προβολής και είναι υπεύθυνος για τη διαχείριση της αλληλεπίδρασης μεταξύ των δύο. Συνήθως περιέχει

χειριστές συμβάντων και άλλη λογική που ανταποκρίνονται στην είσοδο του χρήστη και ενημερώνουν το μοντέλο (Model) και την προβολή (View) ανάλογα με τις ανάγκες.

CSS (Cascading Style Sheets) : Είναι μια γλώσσα που χρησιμοποιείται για την περιγραφή του οπτικού στυλ και της εμφάνισης των διεπαφών χρήστη στην JavaFX. Επιτρέπει τον διαχωρισμό, τον σχεδιασμό και τη διάταξη των στοιχείων του UI από τη λειτουργικότητα και τη συμπεριφορά τους. Διαμορφώνει διάφορα στοιχεία, όπως κουμπιά, ετικέτες, πεδία κειμένου και άλλα. Καθορίζει ιδιότητες όπως το χρώμα, τη γραμματοσειρά, το μέγεθος, το padding και την τοποθέτηση αυτών των στοιχείων. Επίσης, ακολουθεί παρόμοια σύνταξη με την CSS για την ανάπτυξη ιστοσελίδων, αλλά με ορισμένες ειδικές για το JavaFX επιλογείς και ιδιότητες. Εφαρμόζεται σε στοιχεία απευθείας μέσω κώδικα, είτε χρησιμοποιώντας ένα εξωτερικό φύλλο στυλ. Εν κατακλείδι, βοηθάει στην δημιουργία πιο οπτικά ελκυστικών και συνεπή UI, και μπορεί να διευκολύνει τη συντήρηση και την ενημέρωση της εμφάνισης της εφαρμογής καθώς γίνονται αλλαγές με την πάροδο του χρόνου.

Ανακεφαλαιώνοντας, σε μια εφαρμογή JavaFX που δημιουργείται με το NetBeans, η δομή του έργου περιλαμβάνει συνήθως ξεχωριστούς καταλόγους για τα στοιχεία Model, View και Controller, καθώς και άλλους καταλόγους για πόρους όπως εικόνες και CSS. Επιπλέον, το IDE παρέχει εργαλεία για την εύκολη δημιουργία και επεξεργασία αρχείων FXML, καθώς και για τη διαχείριση των εξαρτήσεων και των ρυθμίσεων κατασκευής του έργου. [17]

Όμως, για την δημιουργία των αρχείων FXML που χρησιμοποιούνται στην εφαρμογή που θα αναλυθεί περεταίρω στο επόμενο κεφάλαιο, έγινε χρήση ενός προεραιτικού εργαλείου. Αυτό το εργαλείο είναι το Scene Builder.

Το Scene Builder είναι ένα εργαλείο λογισμικού που επιτρέπει τη δημιουργία διεπαφής χρήστη για τις εφαρμογές JavaFX χρησιμοποιώντας μια οπτική διεπαφή drag-and-drop. Είναι εφικτή η δημιουργία γραφικών διεπαφών χρήστη (GUI), σύροντας και αποθέτοντας στοιχεία UI, όπως κουμπιά, ετικέτες και πεδία κειμένου, σε έναν καμβά. Μπορείτε επίσης να προσθέσετε λειτουργικότητα στα στοιχεία UI συνδέοντάς τα με κώδικα γραμμένο σε Java. Η χρήση του μπορεί να εξοικονομήσει χρόνο και προσπάθεια σε σύγκριση με τη χειροκίνητη υλοποίηση κώδικα των GUI από το μηδέν. Μπορεί επίσης να διευκολύνει την οπτικοποίηση του UI κατά τη δημιουργία, αφού υπάρχει η δυνατότητα να επιτρέπει στον προγραμματιστή να δει πώς θα φαίνονται και πως θα αλληλεπιδρούν τα στοιχεία μεταξύ τους σε πραγματικό χρόνο.

2.2.3 Βάση δεδομένων SQLite

Το SQLite είναι ένα δημοφιλές ελαφρύ και χωρίς διακομιστή σύστημα διαχείρισης βάσεων δεδομένων που προσφέρει αρκετά πλεονεκτήματα για ορισμένες περιπτώσεις χρήσης. Είναι μια συμπαγής βιβλιοθήκη που απαιτεί ελάχιστη εγκατάσταση και διαμόρφωση, καθιστώντας την καλή επιλογή για μικρότερα έργα ή εφαρμογές που πρέπει να αναπτυχθούν γρήγορα και εύκολα. Δεν απαιτεί ξεχωριστή διεργασία διακομιστή ή ειδικό μηχάνημα διακομιστή, γεγονός που καθιστά εύκολη τη ρύθμιση και τη χρήση της σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των επιτραπέζιων και κινητών συσκευών.

Επίσης λειτουργεί σε πολλά λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, macOS, Linux και άλλων. Μπορεί να ενσωματωθεί σε άλλες εφαρμογές, επιτρέποντάς τους να αποθηκεύουν και να διαχειρίζονται δεδομένα τοπικά, χωρίς να απαιτείται ξεχωριστός διακομιστής βάσεων δεδομένων. Έχει σχεδιαστεί για να είναι γρήγορο και αποδοτικό, με μικρό αποτύπωμα μνήμης, γεγονός που το καθιστά ιδανικό για εφαρμογές που πρέπει να διαχειρίζονται γρήγορα μεγάλες ποσότητες δεδομένων. Έχει αποδεδειγμένη αξιοπιστία και σταθερότητα, με μια ενεργή κοινότητα ανάπτυξης που παρέχει συνεχή υποστήριξη και συντήρηση. Είναι συμβατή με πολλές γλώσσες προγραμματισμού, όπως η C/C++, η Python, η Java και άλλες, γεγονός που την καθιστά δημοφιλή επιλογή για πολλούς διαφορετικούς τύπους εφαρμογών.

ΕΙΚΟΝΑ 16 : SQLITE (ΠΗΓΗ : HTTPS://EL.WIKIPEDIA.ORG/WIKI/SQLITE)

Πιο συγκεκριμένα για την γλώσσα προγραμματισμού Java υπάρχουν αρκετές διαθέσιμες βιβλιοθήκες που επιτρέπει την σύνδεση SQLite με την Java, συμπεριλαμβανομένων των SQLite JDBC, Xerial SQLite JDBC και SQLite4Java. Αυτές οι βιβλιοθήκες παρέχουν στους προγραμματιστές ένα API για την αλληλεπίδραση με τη βάση δεδομένων SQLite χρησιμοποιώντας κώδικα Java.

Για να χρησιμοποιήσετε το SQLite στη Java, πρέπει πρώτα να κατεβάσετε τη βιβλιοθήκη SQLite JDBC. Μόλις εγκαταστήσετε το πρόγραμμα οδήγησης, μπορείτε να δημιουργήσετε μια νέα βάση δεδομένων SQLite χρησιμοποιώντας κώδικα Java ή να συνδεθείτε σε μια υπάρχουσα βάση δεδομένων.

Μπορείτε να εκτελέσετε εντολές SQL για να δημιουργήσετε πίνακες, να εισαγάγετε δεδομένα, να ενημερώσετε δεδομένα, να διαγράψετε δεδομένα και να αναζητήσετε δεδομένα από τη βάση δεδομένων. Η βιβλιοθήκη SQLite JDBC παρέχει διάφορες μεθόδους για την εκτέλεση εντολών SQL, όπως οι executeQuery(), executeUpdate() και execute().

Ένα από τα πλεονεκτήματα της χρήσης του SQLite στη Java είναι η απλότητά του. Δεδομένου ότι πρόκειται για μία βάση δεδομένων βασισμένη σε αρχεία. Άρα, δεν χρειάζεται να δημιουργήσετε έναν διακομιστή ή ένα πολύπλοκο περιβάλλον βάσης δεδομένων. Μπορείτε απλώς να συμπεριλάβετε τη βιβλιοθήκη SQLite JDBC στο πειβάλλον ανάπτυξης της εφαρμογής Java και να αρχίσετε να τη χρησιμοποιείτε. [18]

Συνολικά, η SQLite είναι ένα ισχυρό και ευέλικτο σύστημα βάσεων δεδομένων που μπορεί να αποτελέσει εξαιρετική επιλογή για έργα που απαιτούν μια ελαφριά, χωρίς διακομιστή και διαπλατφορμική λύση βάσεων δεδομένων. Επίσης μπορεί εύκολα να ενσωματωθεί με γλώσσες προγραμματισμού όπως η Java.

2.2 Διαγράμματα UML και ER

Τα διαγράμματα Unified Modeling Language (UML) και Entity-Relationship (ER) είναι δύο τύποι διαγραμμάτων που χρησιμοποιούνται συνήθως στη μηχανική λογισμικού και στο σχεδιασμό βάσεων δεδομένων, αντίστοιχα.

Η UML είναι μια γραφική γλώσσα που έχει ως σκοπό τη μοντελοποίηση και το σχεδιασμό συστημάτων λογισμικού. Χρησιμοποιείται για την απεικόνιση, τον καθορισμό, την κατασκευή και την τεκμηρίωση των διαφορετικών στοιχείων ενός συστήματος λογισμικού, συμπεριλαμβανομένων των κλάσεων, των αντικειμένων, των στοιχείων και των σχέσεών τους. Τα διαγράμματα UML μπορούν να χρησιμοποιηθούν για να περιγράψουν τις στατικές και δυναμικές πτυχές ενός συστήματος και βοηθούν τους προγραμματιστές να κατανοήσουν τη σχεδίαση και τη λειτουργικότητα του συστήματος.

Από την άλλη πλευρά, τα διαγράμματα ER χρησιμοποιούνται για το σχεδιασμό και τη μοντελοποίηση βάσεων δεδομένων. Τα διαγράμματα ER αντιπροσωπεύουν οντότητες και τις σχέσεις μεταξύ τους. Οι οντότητες είναι αντικείμενα ή έννοιες που υπάρχουν ανεξάρτητα και έχουν μια μοναδική ταυτότητα. Οι σχέσεις περιγράφουν πώς σχετίζονται οι οντότητες μεταξύ τους. Τα διαγράμματα ER βοηθούν στην οπτικοποίηση της δομής μίας βάσης δεδομένων και των σχέσεων μεταξύ των διαφορετικών πινάκων της. Τα διαγράμματα ER χρησιμοποιούνται για το σχεδιασμό και την υλοποίηση βάσεων δεδομένων και συμβάλλουν στη διασφάλιση ότι τα δεδομένα είναι δομημένα και οργανωμένα με τρόπο που να υποστηρίζει τις ανάγκες της εφαρμογής ή του συστήματος. [19],[20]

Συνοπτικά, τα διαγράμματα UML χρησιμοποιούνται για τη μοντελοποίηση συστημάτων λογισμικού και των στοιχείων τους, ενώ τα διαγράμματα ER χρησιμοποιούνται για τη μοντελοποίηση βάσεων δεδομένων και των οντοτήτων και των σχέσεών τους. Και οι δύο τύποι διαγραμμάτων είναι σημαντικά εργαλεία για το σχεδιασμό λογισμικού, βάσεων δεδομένων και βοηθούν στην οπτικοποίηση, την επικοινωνία της δομής και των σχέσεων του συστήματος ή της βάσης δεδομένων που σχεδιάζεται.

2.2.1 Δέντρα αποφάσεων και διάγραμμα Visual Prolog

Τα δέντρα αποφάσεων είναι ένας τύπος αλγορίθμου μηχανικής μάθησης με επίβλεψη που χρησιμοποιείται τόσο για εργασίες ταξινόμησης όσο και παλινδρόμησης. Ο αλγόριθμος δημιουργεί ένα δενδροειδές μοντέλο αποφάσεων και των πιθανών συνεπειών τους με βάση τα χαρακτηριστικά των δεδομένων εισόδου. Κάθε εσωτερικός κόμβος αντιπροσωπεύει μία απόφαση για ένα χαρακτηριστικό εισόδου, κάθε κλάδος αντιπροσωπεύει το αποτέλεσμα της απόφασης και κάθε κόμβος φύλλου αντιπροσωπεύει την ετικέτα κλάσης ή την τιμή της εργασίας παλινδρόμησης.

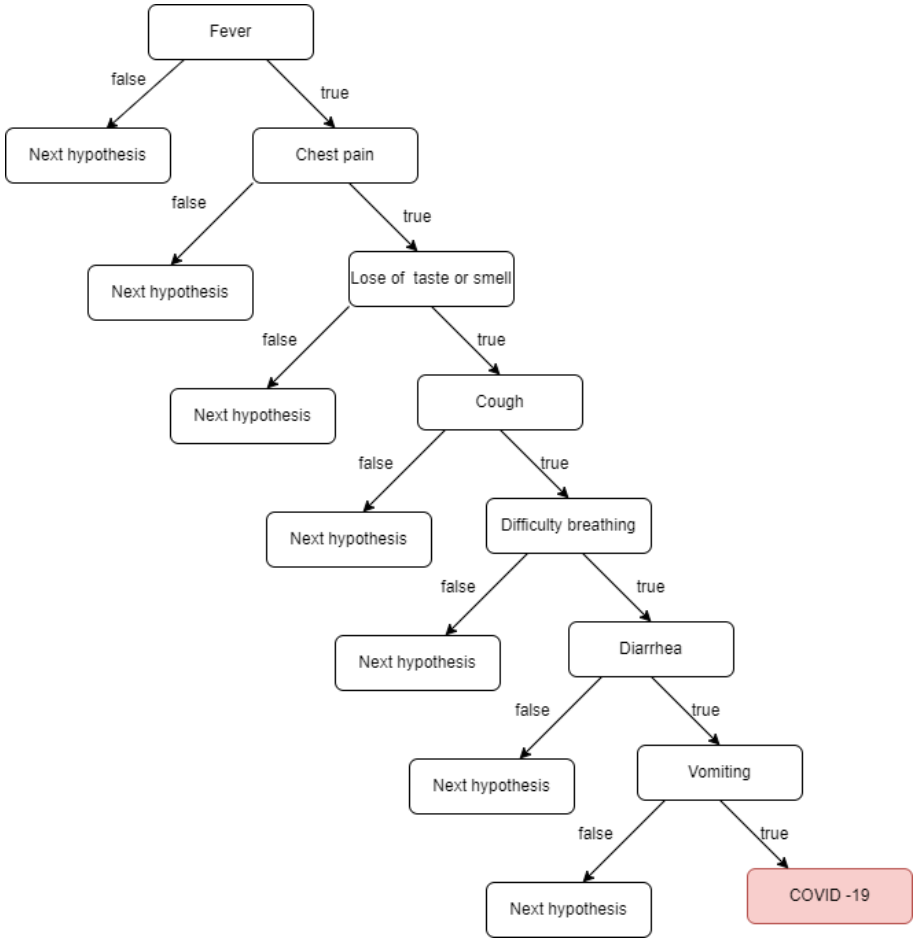
Δημιουργείται μέσω μιας αναδρομικής διαδικασίας που ονομάζεται αναδρομική κατάτμηση. Σε κάθε κόμβο, ο αλγόριθμος επιλέγει το καλύτερο χαρακτηριστικό για να χωρίσει τα δεδομένα σε υποσύνολα. Το καλύτερο χαρακτηριστικό επιλέγεται με βάση ένα κριτήριο που μεγιστοποιεί το κέρδος πληροφορίας ή ελαχιστοποιεί την ακαθαρσία των υποσυνόλων δεδομένων. Τα πιο συχνά χρησιμοποιούμενα κριτήρια για εργασίες ταξινόμησης είναι ο συντελεστής Gini και το κέρδος πληροφορίας, ενώ για εργασίες παλινδρόμησης χρησιμοποιείται το μέσο τετραγωνικό σφάλμα ή το μέσο απόλυτο σφάλμα.

Ο αλγόριθμος δέντρου απόφασης συνεχίζει να διαιρεί τα δεδομένα σε υποσύνολα μέχρι να πληρούνται τα κριτήρια διακοπής, όπως ένα μέγιστο βάθος δέντρου, ένας ελάχιστος αριθμός δειγμάτων ανά κόμβο φύλλου ή όταν όλα τα δείγματα σε έναν κόμβο φύλλου ανήκουν στην ίδια κλάση ή έχουν την ίδια τιμή παλινδρόμησης. Το δέντρο απόφασης που προκύπτει μπορεί να ερμηνευτεί για να αποκτηθούν γνώσεις σχετικά με τις σχέσεις μεταξύ των χαρακτηριστικών εισόδου και της μεταβλητής στόχου.

Έχουν πολλά πλεονεκτήματα, συμπεριλαμβανομένης της ικανότητάς τους να χειρίζονται τόσο κατηγορικά όσο και αριθμητικά χαρακτηριστικά, της ερμηνευτικότητάς τους και της ικανότητάς τους να αποτυπώνουν μη γραμμικές σχέσεις μεταξύ των χαρακτηριστικών και της μεταβλητής στόχου. Ωστόσο, μπορεί να είναι ευαίσθητα σε μικρές μεταβολές στα δεδομένα εισόδου και μπορούν εύκολα να υπερπροσαρμοστούν εάν δεν κλαδευτούν σωστά.

Τέλος τα δέντρα αποφάσεων μπορούν να απεικονιστούν με τη χρήση ενός δενδροειδούς διαγράμματος, όπου κάθε κόμβος αντιπροσωπεύει μια απόφαση και κάθε κλάδος αντιπροσωπεύει ένα αποτέλεσμα. Το διάγραμμα μπορεί να βοηθήσει τους χρήστες να κατανοήσουν πώς ο αλγόριθμος λαμβάνει αποφάσεις και πώς τα χαρακτηριστικά εισόδου επηρεάζουν την έξοδο. [21]

Παρακάτω παρατίθεται ένα δέντρο αποφάσεων, βασισμένο στον κώδικα της εφαρμογής.



ΕΙΚΟΝΑ 17 : ΠΑΡΑΔΕΙΓΜΑ ΔΕΝΤΡΟΥ ΑΠΟΦΑΣΕΩΝ

Ο κώδικας του παραπάνω δέντρου είναι :

```
is_disease('Covid-19') :-
    positive('have', 'fever?'),
    positive('feel', 'chest pain?'),
    positive('have', 'lose of taste or smell?'),
    positive('have', 'cough?'),
    positive('feel', 'difficulty breathing?'),
    positive('have', 'diarrhea?'),
    positive('have', 'vomiting?').
```

Ο συγκεκριμένος κανόνας είναι ένα λογικό κατηγόρημα σε ένα δέντρο αποφάσεων που βοηθά να προσδιοριστεί αν ένα άτομο έχει Covid-19. Ο κανόνας δηλώνει ότι ένα άτομο θεωρείται ότι έχει Covid-19 εάν απαντήσει θετικά σε ένα σύνολο συμπτωμάτων και σημείων που σχετίζονται με την ασθένεια.

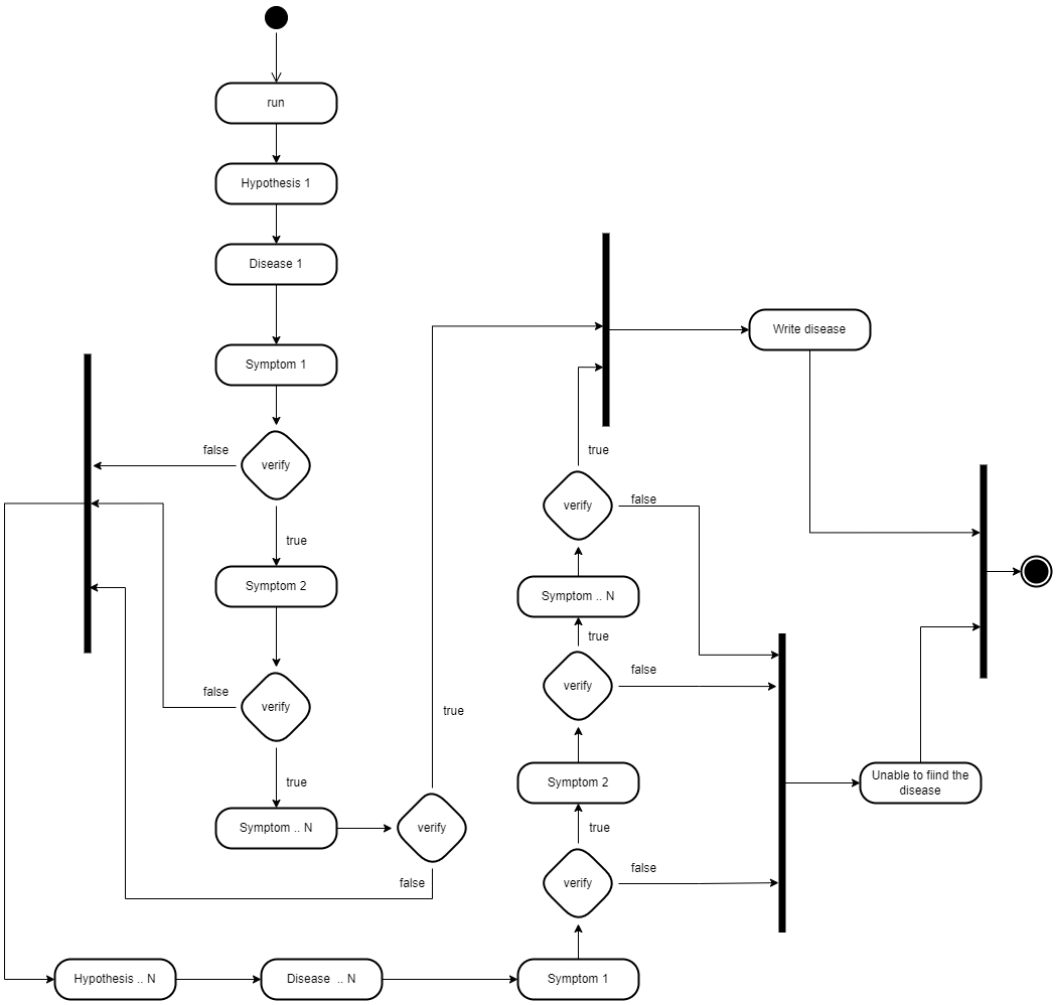
Το κατηγόρημα "is_disease('Covid-19')" είναι το συμπέρασμα ή η υπόθεση αυτού του κανόνα, το οποίο υποδεικνύει ότι εάν ένα άτομο παρουσιάζει το συγκεκριμένο σύνολο συμπτωμάτων, είναι πιθανό να έχει Covid-19.

Ο κανόνας περιλαμβάνει ένα σύνολο συνθηκών ή ερωτήσεων, καθεμία από τις οποίες αντιστοιχεί σε ένα σύμπτωμα ή σημείο που συνήθως συνδέεται με το Covid-19. Αυτές οι συνθήκες εκφράζονται με τη χρήση του κατηγορήματος "θετικό", το οποίο αναμένει δύο ορίσματα: το πρώτο όρισμα είναι μια ερώτηση ναι ή όχι σχετικά με ένα συγκεκριμένο σύμπτωμα και το δεύτερο όρισμα είναι η απάντηση που δίνει το άτομο.

Εάν ένα άτομο απαντήσει θετικά σε όλες τις ερωτήσεις, τότε το κατηγόρημα "is_disease('Covid-19')" αξιολογείται ως αληθές, υποδεικνύοντας ότι το άτομο πιθανώς έχει Covid-19. Ωστόσο, εάν ένα άτομο απαντήσει αρνητικά σε οποιαδήποτε από τις ερωτήσεις, τότε ο κανόνας προχωρά στην επόμενη υπόθεση ή διακλάδωση στο δέντρο αποφάσεων, η οποία μπορεί να είναι άλλος κανόνας ή συνθήκη.

Στην ουσία, ο κανόνας αυτός είναι ένα απλό δέντρο αποφάσεων που βοηθά στον εντοπισμό ατόμων που ενδέχεται να έχουν μολυνθεί με το Covid-19 με βάση ένα σύνολο κοινών συμπτωμάτων και σημείων.

Παρακάτω παρατίθεται το διάγραμμα δραστηριοτήτων, βασισμένο στον κώδικα της εφαρμογής.



ΕΙΚΟΝΑ 18 : ΔΙΑΓΡΑΜΜΑ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ (VISUAL PROLOG)

Το διάγραμμα δραστηριοτήτων αναπαριστά τη διαδικασία διάγνωσης μιας ασθένειας. Με βάση τα συμπτώματα που υπάρχουν ήδη αποθηκευμένα στη μνήμη. Έτσι, ο ασθενής καλείτε να απαντήσει με Ναι ή Οχι στις ερωτήσεις της εφαρμογής.

Το διάγραμμα ξεκινά με έναν αρχικό κόμβο και στη συνέχεια εισέρχεται σε έναν κόμβο απόφασης, όπου το πρόγραμμα ελέγχει αν τα δοσμένα συμπτώματα είναι θετικά ή αρνητικά. Εάν τα συμπτώματα είναι θετικά, το πρόγραμμα μεταβαίνει στη θετική ρήτρα και ελέγχει εάν υπάρχει ταύτιση με γνωστή ασθένεια. Εάν υπάρχει ταύτιση, το πρόγραμμα εμφανίζει το όνομα της ασθένειας χρησιμοποιώντας ένα πλαίσιο μηνυμάτων και στη συνέχεια μεταβαίνει στη ρήτρα. Εάν δεν υπάρχει ταύτιση, εμφανίζει μήνυμα στον χρήστη πως δεν βρέθηκε κανένα αποτέλεσμα.

Εάν τα συμπτώματα είναι αρνητικά, το πρόγραμμα μεταβαίνει σε επόμενη υπόθεση για να θέσει στον ασθενή περισσότερες ερωτήσεις για τον περαιτέρω προσδιορισμό της διάγνωσης.

Αυτός ο κύκλος συνεχίζεται μέχρι να βρεθεί μια θετική διάγνωση ή να απαντηθούν όλες οι ερωτήσεις με αρνητική απάντηση, οπότε το πρόγραμμα εμφανίζει ένα μήνυμα σφάλματος.

Το διάγραμμα τελειώνει με έναν τελικό κόμβο που υποδεικνύει το τέλος του προγράμματος.

2.2.2 Διαγράμματα Java

Το διάγραμμα κλάσεων είναι ένας τύπος διαγράμματος UML (Unified Modeling Language) που αναπαριστά τη δομή ενός συστήματος λογισμικού με βάση τις κλάσεις, τα χαρακτηριστικά τους, τις μεθόδους και τις σχέσεις με άλλες κλάσεις. Είναι μια οπτική αναπαράσταση της στατικής δομής του συστήματος και χρησιμοποιείται για τη μοντελοποίηση των κλάσεων και των αλληλεπιδράσεών τους.

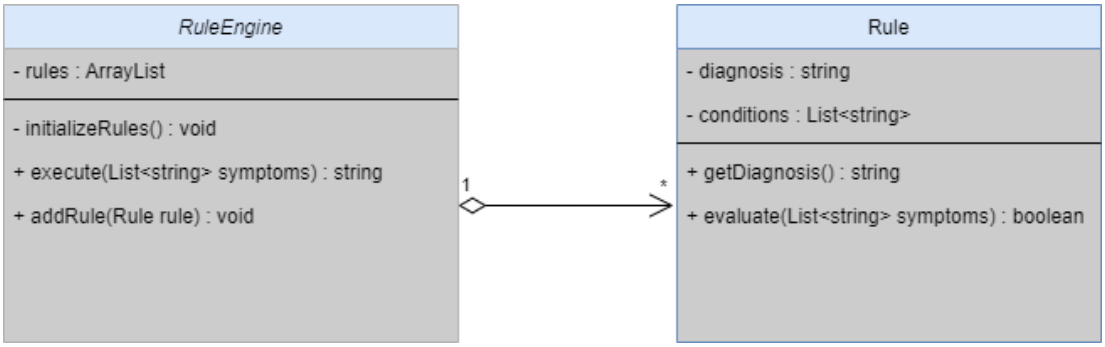
Σε ένα διάγραμμα κλάσεων, κάθε κλάση αναπαρίσταται ως ορθογώνιο με το όνομα της κλάσης στην κορυφή. Τα χαρακτηριστικά της κλάσης (μεταβλητές) παρατίθενται κάτω από το όνομα και οι μέθοδοι της κλάσης (συναρτήσεις) παρατίθενται κάτω από τα χαρακτηριστικά. Οι σχέσεις μεταξύ των κλάσεων αναπαρίστανται με βέλη, τα οποία υποδεικνύουν την κατεύθυνση της σχέσης.

Ορισμένοι συνηθισμένοι τύποι σχέσεων σε ένα διάγραμμα κλάσεων περιλαμβάνουν :

- Κληρονομικότητα: Μια σχέση μεταξύ μιας γονικής κλάσης (υπερκλάσης) και μιας κλάσης-παιδιού (υποκλάσης). Το βέλος δείχνει από την υποκλάση στην υπερκλάση, υποδεικνύοντας ότι η υποκλάση κληρονομεί τα χαρακτηριστικά και τις μεθόδους της υπερκλάσης.
- Συσχέτιση: Μια σχέση μεταξύ δύο κλάσεων, όπου μια κλάση χρησιμοποιεί μια περίπτωση μιας άλλης κλάσης. Το βέλος δείχνει από την κλάση που χρησιμοποιεί την άλλη κλάση στη χρησιμοποιούμενη κλάση.
- Συγκέντρωση: Μια σχέση μεταξύ δύο κλάσεων, όπου μια κλάση περιέχει περιπτώσεις μιας άλλης κλάσης. Το βέλος δείχνει από την κλάση που περιέχει στην κλάση που περιέχεται.
- Σύνθεση: Ένας ειδικός τύπος συνάθροισης όπου η κλάση που περιέχεται δεν μπορεί να υπάρξει χωρίς την κλάση που περιέχει. Το βέλος δείχνει από την κλάση που περιέχει στην κλάση που περιέχεται και αναπαρίσταται ως ένα γεμάτο διαμάντι.

Τα διαγράμματα κλάσεων μπορούν να χρησιμοποιηθούν κατά τη φάση σχεδιασμού της ανάπτυξης λογισμικού για τον προσδιορισμό των κλάσεων που απαιτούνται για το σύστημα και των σχέσεών τους. Μπορούν επίσης να χρησιμοποιηθούν για την επικοινωνία του σχεδιασμού του συστήματος σε άλλους προγραμματιστές και ενδιαφερόμενους. Τα διαγράμματα κλάσεων αποτελούν σημαντικό εργαλείο για τον αντικειμενοστραφή προγραμματισμό και σχεδιασμό και χρησιμοποιούνται συνήθως στην ανάπτυξη εφαρμογών λογισμικού.

Παρακάτω παρετείθεντε δύο διαγράμματα κλάσεων, βασισμένα στον κώδικα της εφαρμογής.



ΕΙΚΟΝΑ 19 : ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΗΣ (CLASS DIAGRAM)

Η κλάση RuleEngine είναι μια κλάση κοντέινερ που διαχειρίζεται μια συλλογή αντικειμένων Rule.

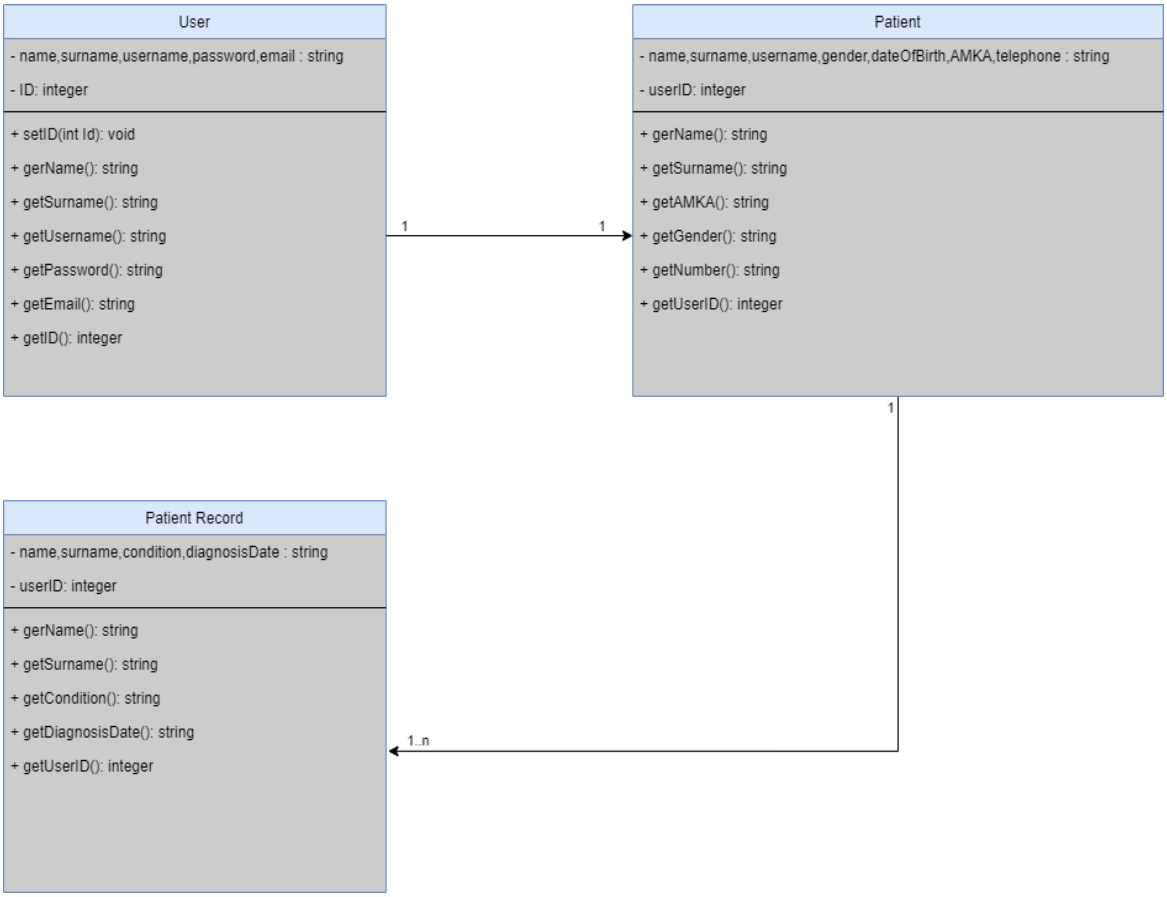
Η κλάση Rule αναπαριστά έναν μεμονωμένο κανόνα που χρησιμοποιείται για την αξιολόγηση ενός συνόλου συνθηκών και την ανάληψη μιας ενέργειας με βάση το αποτέλεσμα.

Η κλάση RuleEngine έχει μια συσχέτιση με την κλάση Rule, υποδεικνύοντας ότι διαθέτει μια συλλογή αντικειμένων Rule. Αυτή η συσχέτιση αναπαρίσταται από μια σχέση συγκέντρωσης με δυνατότητα πλοήγησης στο διάγραμμα κλάσεων, με ένα βέλος σε σχήμα διαμαντιού που δείχνει από την κλάση RuleEngine στην κλάση Rule.

Η κλάση Rule δεν έχει καμία συσχέτιση με την κλάση RuleEngine, καθώς διαχειρίζεται από την κλάση RuleEngine και δεν χρειάζεται να την αναφέρει άμεσα.

Συνολικά, οι κλάσεις RuleEngine και Rule συνεργάζονται για να παρέχουν ένα διαγνωστικό σύστημα βασισμένο σε κανόνες. Η κλάση RuleEngine διαχειρίζεται μια συλλογή αντικειμένων Rule και τα αντικείμενα Rule περιέχουν τις συγκεκριμένες συνθήκες και τη σχετική διάγνωση για κάθε κανόνα. Όταν δίνεται ένα σύνολο συμπτωμάτων ως είσοδος, η

κλάση RuleEngine αξιολογεί κάθε αντικείμενο Rule για να καθορίσει ποιος κανόνας ταιριάζει με τα συμπτώματα και επιστρέφει την αντίστοιχη διάγνωση.



ΕΙΚΟΝΑ 20 : ΔΙΑΓΡΑΜΜΑ ΣΥΣΧΕΤΙΣΕΩΝ (ER)

Η κλάση User αντιπροσωπεύει έναν χρήστη του συστήματος και έχει τα ακόλουθα χαρακτηριστικά: όνομα, επώνυμο, όνομα χρήστη, κωδικός πρόσβασης και email. Αυτή η κλάση αντιπροσωπεύει έναν γενικό χρήστη και δεν είναι εξειδικευμένη για να αντιπροσωπεύει έναν συγκεκριμένο τύπο χρήστη στο σύστημα.

Η κλάση Patient αντιπροσωπεύει έναν συγκεκριμένο τύπο χρήστη στο σύστημα, ο οποίος είναι ένας ασθενής. Αυτή η κλάση έχει τα ακόλουθα χαρακτηριστικά: userId (το οποίο είναι ξένο κλειδί για την κατηγορία User), όνομα, επώνυμο, φύλο, ημερομηνία γέννησης , αριθμός κοινωνικής ασφάλισης και τηλέφωνο. Αυτή η κλάση κληρονομεί όλα τα χαρακτηριστικά και τις μεθόδους από την κατηγορία User, καθώς όλοι οι ασθενείς είναι επίσης χρήστες.

Η κλάση PatientRecord αντιπροσωπεύει έναν ιατρικό φάκελο για έναν ασθενή και έχει τα ακόλουθα χαρακτηριστικά: όνομα (του ασθενούς), επώνυμο (του ασθενούς), κατάσταση (η ιατρική πάθηση που έχει) και ημερομηνία διάγνωσης της πάθησης. Αυτή η κατηγορία έχει

σχέση πολλά προς ένα με την κατηγορία Patient, καθώς κάθε ασθενής μπορεί να έχει πολλαπλούς ιατρικούς φακέλους, αλλά κάθε ιατρικός φάκελος ανήκει σε έναν μόνο ασθενή.

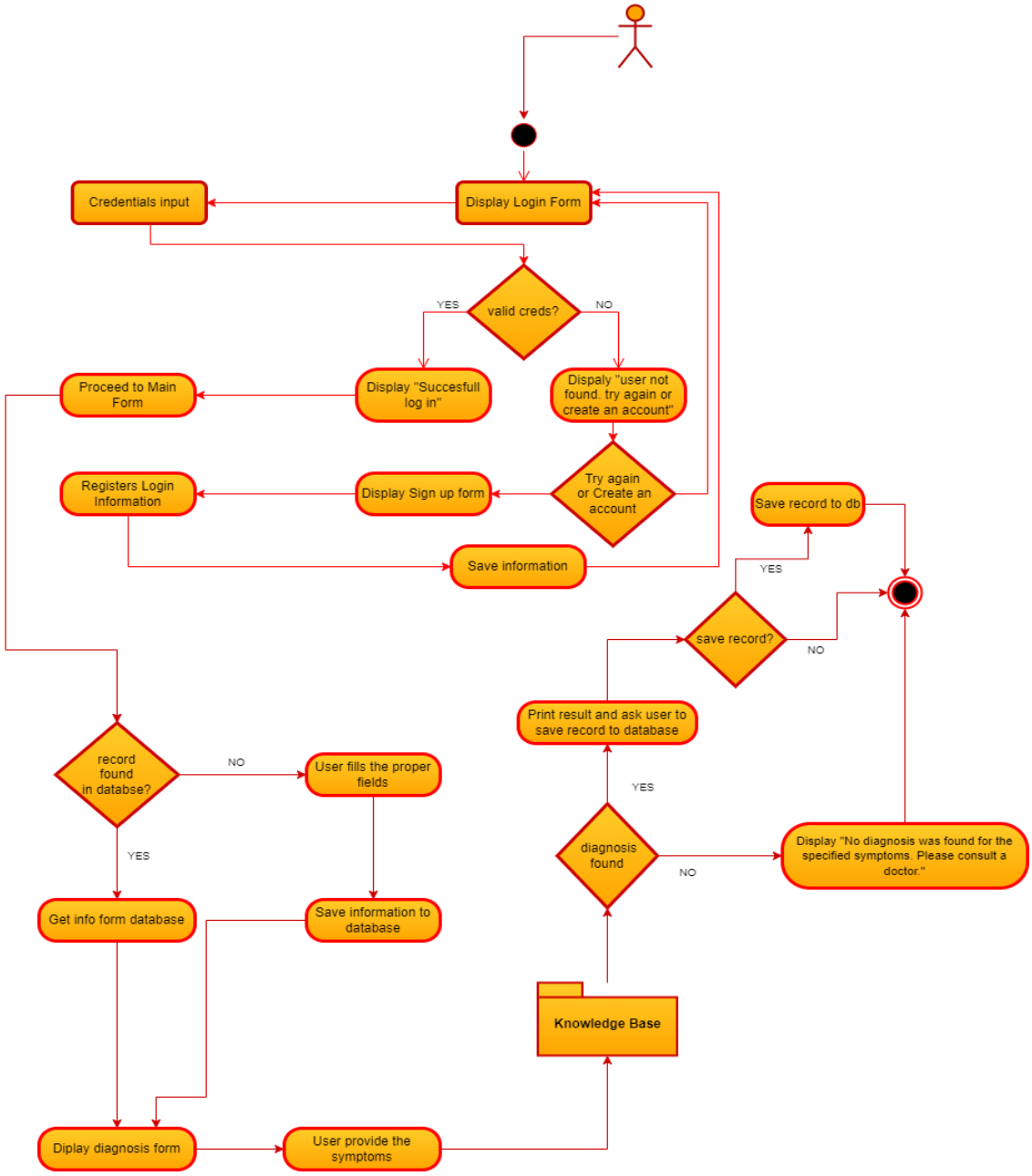
Το διάγραμμα ER για αυτές τις κατηγορίες θα εμφανίσει τις ακόλουθες σχέσεις.

Η κλάση User θα έχει σχέση ένας προς έναν με την κατηγορία Patient, καθώς κάθε ασθενής είναι χρήστης, αλλά δεν είναι όλοι οι χρήστες ασθενείς.

Η κατηγορία Patient θα έχει σχέση ένα προς πολλά με την κατηγορία PatientRecord, καθώς κάθε ασθενής μπορεί να έχει πολλαπλούς ιατρικούς φακέλους, αλλά κάθε ιατρικός φάκελος ανήκει σε έναν μόνο ασθενή.

Συνολικά, αυτές οι τρεις κατηγορίες αντιπροσωπεύουν ένα βασικό σύστημα για τη διαχείριση των ασθενών και των ιατρικών τους αρχείων. Η κλάση User παρέχει τα βασικά χαρακτηριστικά και μεθόδους για όλους τους χρήστες, ενώ η κλάση Patient επεκτείνει την κλάση User για να παρέχει πρόσθετα χαρακτηριστικά και μεθόδους ειδικά για ασθενείς. Τέλος, η κλάση PatientRecord αντιπροσωπεύει τα ιατρικά αρχεία που σχετίζονται με κάθε ασθενή, με μια σχέση ένα προς πολλά με την κατηγορία Patient.

Παρακάτω παρατίθεται το διάγραμμα δραστηριοτήτων, βασισμένο στον κώδικα της εφαρμογής.



ΕΙΚΟΝΑ 21 : ΔΙΑΓΡΑΜΜΑ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ (JAVA)

Ακολουθεί ανάλυση των διαφόρων βημάτων στο διάγραμμα :

- Εκκίνηση εφαρμογής: Αυτή είναι η αρχή της εφαρμογής.
- Φόρμα σύνδεσης: Στον χρήστη παρουσιάζεται μια φόρμα σύνδεσης όπου μπορεί να εισάγει το όνομα χρήστη και τον κωδικό πρόσβασής του.
- Εγγραφή: Εάν ο χρήστης δεν έχει εγγραφεί ακόμη, μπορεί να επιλέξει την επιλογή "Εγγραφή" για να δημιουργήσει έναν νέο λογαριασμό.
- Κύρια φόρμα: Μετά τη σύνδεση ή την εγγραφή, ο χρήστης οδηγείται στην κύρια φόρμα όπου μπορεί να έχει πρόσβαση σε διάφορες λειτουργίες της εφαρμογής.

- Συμπλήρωση στοιχείων χρήστη: Εάν είναι η πρώτη φορά που ο χρήστης συνδέεται, θα του ζητηθεί να συμπληρώσει τα προσωπικά του στοιχεία.
- Φόρμα διάγνωσης: Ο χρήστης μπορεί να έχει πρόσβαση στη λειτουργία "Διάγνωση" από την κύρια φόρμα, η οποία τον μεταφέρει στη φόρμα διάγνωσης όπου μπορεί να καταχωρίσει τα συμπτώματά του.
- Εισαγωγή συμπτωμάτων: Ο χρήστης εισάγει τα συμπτώματά του στη φόρμα διάγνωσης.
- Λήψη διάγνωσης: Η εφαρμογή αξιολογεί τα συμπτώματα και παρέχει μια διάγνωση, εάν βρεθεί.
- Αποθήκευση εγγραφής: Εάν βρεθεί διάγνωση, η εφαρμογή προτρέπει τον χρήστη να αποθηκεύσει τον ιατρικό του φάκελο. Εάν το επιλέξει, ο φάκελός του αποθηκεύεται στη βάση δεδομένων.
- Έξοδος: Εάν ο χρήστης επιλέξει να μην συνδεθεί ή να μην εγγραφεί, μπορεί να εξέλθει από την εφαρμογή σε οποιοδήποτε σημείο.

Συνολικά, αυτό το διάγραμμα δραστηριοτήτων δείχνει τα διάφορα βήματα που μπορεί να ακολουθήσει ένας χρήστης κατά τη χρήση αυτής της εφαρμογής. Το διάγραμμα ξεκινά με την είσοδο ή την εγγραφή του χρήστη και στη συνέχεια προχωρά στην κύρια φόρμα όπου μπορεί να έχει πρόσβαση σε διάφορες λειτουργίες της εφαρμογής. Από εκεί, μπορεί να έχει πρόσβαση στη φόρμα διάγνωσης, όπου εισάγει τα συμπτώματά του και λαμβάνει μία διάγνωση, εάν βρεθεί. Εάν βρεθεί διάγνωση, δίνεται η δυνατότητα για αποθήκευση των αποτελεσμάτων στη βάση δεδομένων. Το διάγραμμα ολοκληρώνεται με την έξοδο του χρήστη από την εφαρμογή, εάν επιλέξει να μην κάνει είσοδο ή εγγραφή.

Κεφαλαιο 3ο

Το παρόν κεφάλαιο πραγματεύεται την αναλυτική περιγραφή της δημιουργίας της εφαρμογής, χρησιμοποιώντας μία τμηματική και βήμα προς βήμα ανάλυση των διαφόρων τμημάτων που απαρτίζουν τον κώδικα της εφαρμογής. Στη συνέχεια, θα γίνει παρουσίαση της εφαρμογής με τη χρήση στιγμιοτύπων από την ενδεικτική ροή του χρήστη, εξηγώντας παράλληλα τη λειτουργικότητα που μεσολάβησε και τα διάφορα χαρακτηριστικά που απεικονίζονται σε κάθε στιγμιότυπο, καταλήγοντας στην εξαγωγή αποτελέσματος. Τέλος, θα πραγματοποιηθεί μία συγκριτική ανάλυση μεταξύ των δύο γλωσσών προγραμματισμού που χρησιμοποιήθηκαν για τις διαφορετικές υλοποιήσεις της εφαρμογής. Η ανάλυση θα περιλαμβάνει ομοιότητες και διαφορές των δύο γλωσσών, τους σκοπούς και τη σύνηθη χρήση κάθε γλώσσας, ομοιότητες και διαφοροποιήσεις που παρουσιάστηκαν κατά τη διαδικασία της υλοποίησης, όπως και τον αντίκτυπο της κάθε γλώσσας στη διευκόλυνση και αποτελεσματικότητα της επεξεργασίας των δεδομένων.

3.1 Δημιουργία εφαρμογής Visual Prolog

```
class facts - factdb
    xpositive : (symbol, symbol).
    xnegative : (symbol, symbol) nondeterm.
```

ΕΙΚΟΝΑ 22 : ΟΡΙΣΜΟΣ ΙΔΙΟΤΗΤΩΝ

Στο τμήμα class_facts ορίζονται δύο ιδιότητες, η xpositive και η xnegative. Η xpositive αποθηκεύει το σύμπτωμα και το όνομα της ασθένειας αν το σύμπτωμα είναι θετικό για την ασθένεια. Η xnegative αποθηκεύει το σύμπτωμα και το όνομα της ασθένειας αν το σύμπτωμα είναι αρνητικό για την ασθένεια.

```
clauses
    positive(X, Y) :-
        xpositive(X, Y),
        !.
    positive(X, Y) :-
        not(xnegative(X, Y)),
        question(X, Y).
```

ΕΙΚΟΝΑ 23 : ΚΑΝΟΝΑΣ POESITIVE

Ο κανόνας positive ελέγχει αν ένα σύμπτωμα είναι αληθές (θετικό) για μία ασθένεια. Εάν είναι αληθές τότε ο κανόνας θεωρείτε επιτυχής. Ενώ εάν είναι αρνητικό ρωτάει τον χρήστη με μία ερώτηση σχετική με το σύμπτωμα.

```
clauses
    question(X, Y) :-
        !,
        Reply = answerDialog::ask(string::concat(X, " it ", Y, "\n")),
        % !,
        C = string::charLower(string::frontChar(Reply)),
        remember(X, Y, C).
```

ΕΙΚΟΝΑ 24 : ΚΑΝΟΝΑΣ QUESTION

Ο κανόνας question εμφανίζει ένα πλαίσιο μηνύματος, με μία ερώτηση σχετικά με ένα σύμπτωμα και περιμένει την απάντηση του χρήστη. Στη συνέχεια η απάντηση αποθηκεύεται στη βάση δεδομένων γεγονότων, χρησιμοποιώντας τον κανόνα remember.

```
clauses
    remember(X, Y, Ans) :-
        if Ans = 'y' then
            asserta(xpositive(X, Y))
        elseif Ans = 'n' then
            asserta(xnegative(X, Y)),
            fail
        end if.
```

ΕΙΚΟΝΑ 25 : ΚΑΝΟΝΑΣ REMEMBER

Ο κανόνας remember δέχεται τρία ορίσματα, το σύμπτωμα, την ασθένεια και την απάντηση του χρήστη στην ερώτηση. Εάν η απάντηση είναι “y” (ναι), τότε το σύμπτωμα αποθηκεύεται ως θετικό για την ασθένεια, χρησιμοποιώντας το κατηγόρημα asserta. Εάν η απάντηση είναι “n” (όχι), τότε το σύμπτωμα αποθηκεύεται ως αρνητικό για την ασθένεια, χρησιμοποιώντας το κατηγόρημα asserta και ο κανόνας αποτυγχάνει. Με τον τρόπο αυτό διασφαλίζεται ότι το σύστημα επιστρέφει πίσω και θέτει στον χρήστη μία διαφορετική ερώτηση.

```
clear_facts() :-
    retractFactDb(factDb).
    %_Reply = stdio::readLine().
```

ΕΙΚΟΝΑ 26 : ΚΑΝΟΝΑΣ CLEAR_FACTS

Ο κανόνας clear_facts καθαρίζει τη βάση δεδομένων γεγονότων με την ανάκληση όλων των γεγονότων. Χρησιμοποιώντας το κατηγόρημα retractFactDb.

```
clauses
    run() :-
        disease(X),
        !,
        messageBox::displayNote(X, "The disease having all these symptoms is"),
        clear_facts.

    run() :-
        messageBox::displayError("The disease cannot be determined."),
        clear_facts.
```

ΕΙΚΟΝΑ 27 : ΚΑΝΟΝΑΣ RUN

Ο κανόνας run προσπαθεί να διαγνώσει την ασθένεια με βάση τα συμπτώματα. Ελέγχει κάθε ασθένεια με τη σειρά, καλώντας τον κανόνα disease. Εάν βρεθεί μία ασθένεια, τότε εμφανίζεται ένα πλαίσιο μηνύματος με το όνομα της ασθένειας. Διαφορετικά, εμφανίζεται ένα μήνυμα σφάλματος. Και στις δύο περιπτώσεις, η βάση δεδομένων γεγονότων διαγράφεται με την χρήση του κανόνα clear_facts.

```
clauses
    disease('Covid-19') :-
        is_disease('Covid-19').

    disease('Pharyngitis') :-
        is_disease('Pharyngitis').

    disease('Flu') :-
        is_disease('Flu').

    disease('Common Cold') :-
        is_disease('Common Cold').

    disease('Pneumonia') :-
        is_disease('Pneumonia').

    disease('Menigitis') :-
        is_disease('Menigitis').
```

ΕΙΚΟΝΑ 28 : ΚΑΝΟΝΑΣ DISEASE

Ο κανόνας disease δέχεται ένα όνομα ασθένειας ως όρισμα και ελέγχει αν τα συμπτώματα για την εν λόγω ασθένεια είναι θετικά. Χρησιμοποιώντας τον κανόνα is_disease. Εάν όλα τα συμπτώματα είναι θετικά, τότε η ασθένεια θεωρείται διαγνωσμένη.

```
clauses
    is_disease('Covid-19') :-
        positive('have', 'fever?'),
        positive('feel', 'chest pain?'),
        positive('have', 'lose of taste or smell?'),
        positive('have', 'cough?'),
        positive('feel', 'difficulty breathing?'),
        positive('have', 'diarrhea?'),
        positive('have', 'vomiting?').

    is_disease('Pharyngitis') :-
        positive('have', 'sudden onset of sore throat?'),
        positive('have', 'pain with swallowing?'),
        positive('have', 'fever?'),
        positive('have', 'cough?'),
        positive('have', 'rhinorrea?'),
        positive('have', 'hoarseness?'),
        positive('have', 'oral ulcers?'),
        positive('have', 'conjuctivitis?').

    is_disease('Flu') :-
        positive('have', 'fever?'),
        positive('have', 'cough?'),
        positive('have', 'sore throat?'),
        positive('have', 'runny or stuffy nose?'),
        positive('have', 'muscle or body aches?'),
        positive('have', 'headaches?'),
        positive('have', 'fatigue?').

    is_disease('Common Cold') :-
        positive('have', 'runny or stuffy nose?'),
        positive('have', 'sore throat?'),
        positive('have', 'headaches?').

    is_disease('Pneumonia') :-
        positive('have', 'fever and chills'),
        positive('have', 'cough?'),
        positive('have', 'rappid breathing or difficulty breathing?'),
        positive('have', 'chest pain?').

    is_disease('Menigitis') :-
        positive('have', 'stiff neck?'),
        positive('have', 'fever?'),
        positive('have', 'headache?'),
        positive('have', 'photophobia (eyes being more sensitive to light)?'),
        positive('have', 'confusion?').
```

ΕΙΚΟΝΑ 29 : ΚΑΝΟΝΑΣ IS_DESEASE

Ο κανόνας is_disease δέχεται ένα όνομα ασθένειας ως όρισμα και ελέγχει αν κάθε σύμπτωμα, για την εν λόγω ασθένεια είναι θετικό. Χρησιμοποιώντας τον κανόνα positive.

Συνολικά αυτός ο κώδικας είναι ένα παράδειγμα ενός έμπειρου συστήματος. Βασισμένο σε κανόνες, που χρησιμοποιεί ένα σύνολο κανόνων για την διάγνωση μίας ασθένειας, με βάση ένα σύνολο συμπτωμάτων. Το σύστημα θέτει ερωτήσεις στον χρήστη για να καθορίσει, ποια συμπτώματα υπάρχουν. Στη συνέχεια χρησιμοποιεί ένα σύνολο κανόνων για την διάγνωση της νόσου. [22]

3.2 Δημιουργία εφαρμογής Java

Κλάση Rule : Η κλάση Rule αναπαριστά έναν κανόνα στο σύστημα βασισμένο σε κανόνες για τη διάγνωση ασθενειών με βάση συγκεκριμένα συμπτώματα. Ένα αντικείμενο Rule περιέχει μια διάγνωση (όνομα ασθένειας) και μια λίστα συνθηκών (συμπτώματα που σχετίζονται με την εν λόγω ασθένεια).

```
private final  String diagnosis;
private final  List<String> conditions;

public Rule(String diagnosis, List<String> conditions)
{
    this.diagnosis = diagnosis;
    this.conditions = conditions;
}
```

ΕΙΚΟΝΑ 30 : ΚΑΤΑΣΚΕΥΑΣΤΗΣ RULE

Κατασκευαστής Rule(String diagnosis, List<String> conditions) : Κατασκευαστής που δέχεται ως είσοδο μια διάγνωση (όνομα ασθένειας) και μια λίστα συνθηκών (συμπτώματα που σχετίζονται με την ασθένεια).

```
public String getDiagnosis()
{
    return this.diagnosis;
}
```

ΕΙΚΟΝΑ 31 : ΜΕΘΟΔΟΣ GETDIAGNOSIS

Μέθοδος getDiagnosis() : μέθοδος που επιστρέφει τη διάγνωση που σχετίζεται με τον κανόνα.

```
public boolean evaluate(List<String> symptoms)
{
    var _conditions  = this.conditions.toArray();
    var _symptoms = symptoms.toArray();
    Arrays.sort(a: _symptoms);
    Arrays.sort(a: _conditions);

    return Arrays.deepEquals(a1: _conditions, a2: _symptoms);
}
```

ΕΙΚΟΝΑ 32 : ΜΕΘΟΔΟΣ EVALUATE

Μέθοδος evaluate(List<String> symptoms) : μέθοδος που δέχεται ως είσοδο μια λίστα συμπτωμάτων και επιστρέφει true εάν όλα τα συμπτώματα της λίστας ταιριάζουν με τις συνθήκες που σχετίζονται με την ασθένεια στον κανόνα.

Κλάση RuleEngine : Η κλάση RuleEngine είναι η κύρια κλάση που περιέχει μια λίστα κανόνων και διαθέτει μεθόδους για την αρχικοποίηση των κανόνων, την προσθήκη νέων κανόνων και την εκτέλεση των κανόνων με βάση συγκεκριμένα συμπτώματα.

```
public RuleEngine()
{
    initializeRules();
}
```

ΕΙΚΟΝΑ 33 : ΚΑΤΑΣΚΕΥΑΣΤΗΣ RULEENGINE

Κατασκευαστής RuleEngine() : Κατασκευαστής που αρχικοποιεί τους κανόνες καλώντας τη μέθοδο initializeRules.

```
private void initializeRules()
{
    this.rules.add(new Rule(diagnosis: "Covid-19",
                            conditions: Arrays.asList(a: "Cough", a: "Fever or chills", a: "Headache", a: "Loss of taste or smell")));

    this.rules.add(new Rule(diagnosis: "Common flu",
                            conditions: Arrays.asList(a: "Cough", a: "Fatigue", a: "Headache")));

    this.rules.add(new Rule(diagnosis: "Pneumonia",
                            conditions: Arrays.asList(a: "Cough", a: "Difficulty breathing", a: "Fever or chills", a: "Chest Pain")));
}
```

ΕΙΚΟΝΑ 34 : ΜΕΘΟΔΟΣ INITIALIZERULES

Μέθοδος initializeRules() : Μέθοδος που αρχικοποιεί τη λίστα κανόνων με ένα σύνολο προκαθορισμένων κανόνων (ασθένειες και τα συναφή συμπτώματά τους).

```
public void addRules(Rule rule)
{
    this.rules.add(e: rule);
}
```

ΕΙΚΟΝΑ 35 : ΜΕΘΟΔΟΣ ADDRULES

Μέθοδος addRules(Rule rule) : μέθοδος που δέχεται ένα αντικείμενο Rule ως είσοδο και το προσθέτει στη λίστα κανόνων.

```
public String execute(List<String> symptoms)
{
    for (Rule rule : this.rules)
    {
        if (rule.evaluate(symptoms))
        {
            return rule.getDiagnosis();
        }
    }
    return "unrecognized";
}
```

ΕΙΚΟΝΑ 36 : ΜΕΘΟΔΟΣ EXECUTE

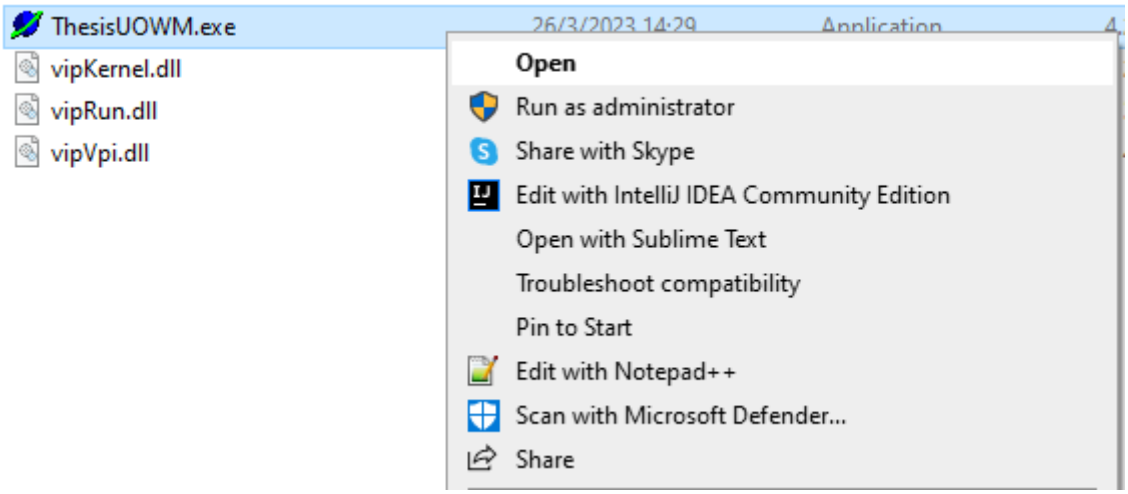
Μέθοδος execute(List<String> symptoms) : μέθοδος που δέχεται μια λίστα συμπτωμάτων ως είσοδο και εκτελεί τους κανόνες με επανάληψη κάθε κανόνα στη λίστα κανόνων και κλήση της μεθόδου evaluate() για να ελέγξει αν τα συμπτώματα που δίνονται ταιριάζουν με τα συμπτώματα που σχετίζονται με την ασθένεια στον συγκεκριμένο κανόνα. Εάν υπάρχει ταύτιση, η μέθοδος επιστρέφει τη διάγνωση που σχετίζεται με τον εν λόγω κανόνα. Εάν κανένας από τους κανόνες δεν ταιριάζει, η μέθοδος επιστρέφει "unrecognized".

Συνολικά, η κλάση Rule παρέχει μεθόδους για τη δημιουργία και την αξιολόγηση ενός μεμονωμένου κανόνα στο σύστημα βασισμένο σε κανόνες, ενώ η κλάση RuleEngine παρέχει μεθόδους για την αρχικοποίηση και την εκτέλεση ενός συνόλου κανόνων στο σύστημα. Μαζί, αυτές οι κλάσεις σχηματίζουν ένα βασικό σύστημα βασισμένο σε κανόνες για τη διάγνωση ασθενειών με βάση δεδομένα συμπτώματα.[23]

3.3 Παρουσίαση εφαρμογής Visual Prolog

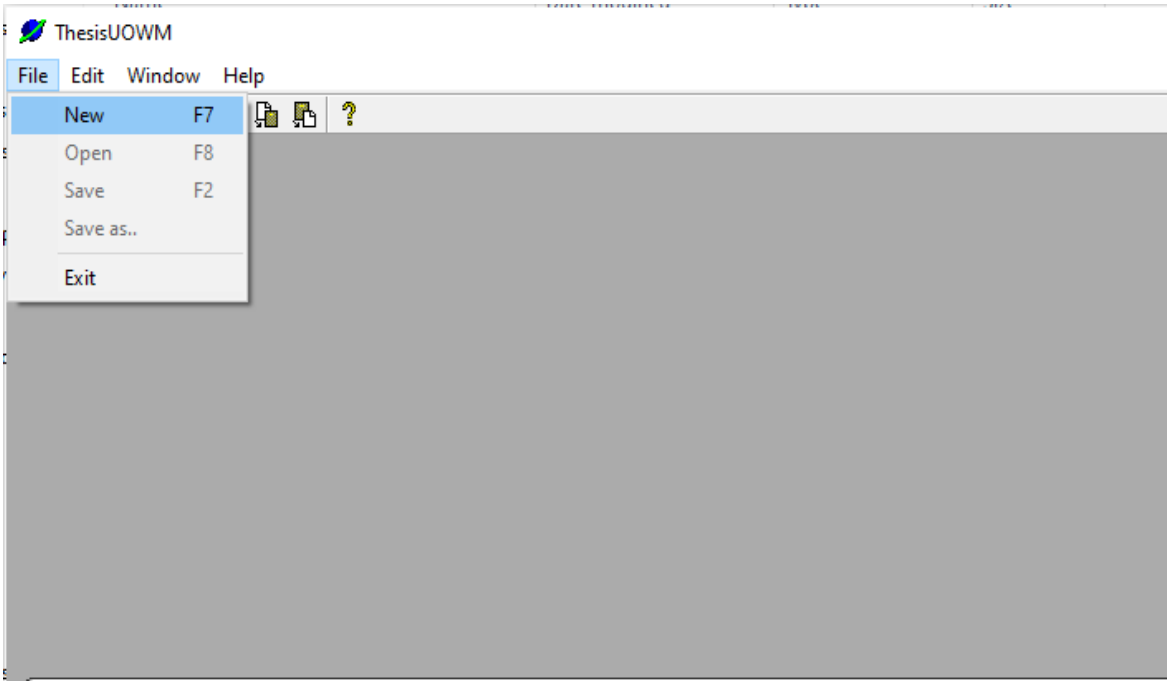
Σε αυτή την ενότητα θα γίνει η παρουσίαση της εφαρμογής, με τα βήματα που θα πρέπει να ακολουθήσει ο χρήστης, στο γραφικό περιβάλλον της εφαρμογής στη Visual Prolog.

Αρχικά ο χρήστης καλείται να εκτελέσει το εκτελέσιμο αρχείο που έχει δημιουργηθεί με το περιβάλλον ανάπτυξης Visual Prolog 10 IDE.



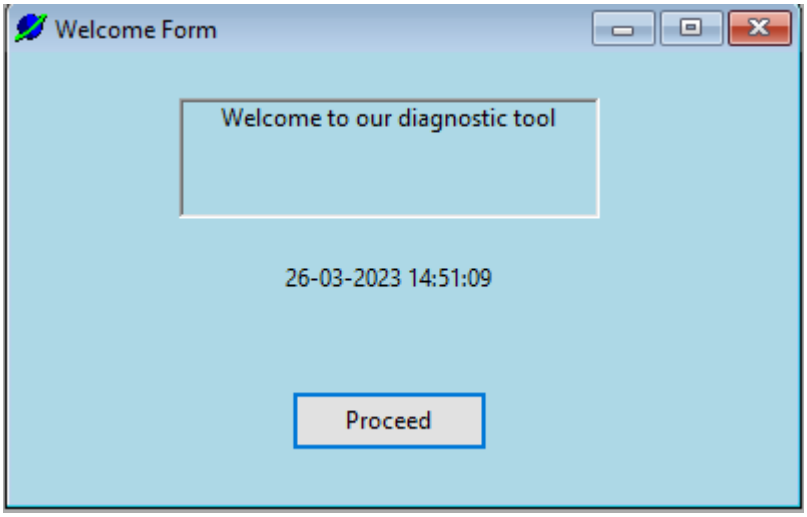
ΕΙΚΟΝΑ 37 : ΠΡΩΤΟ ΒΗΜΑ ΕΚΤΕΛΕΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Αφού, ανοίξει το κυρίως παράθυρο, ο χρήστης θα πρέπει να επιλέξει το File -> New



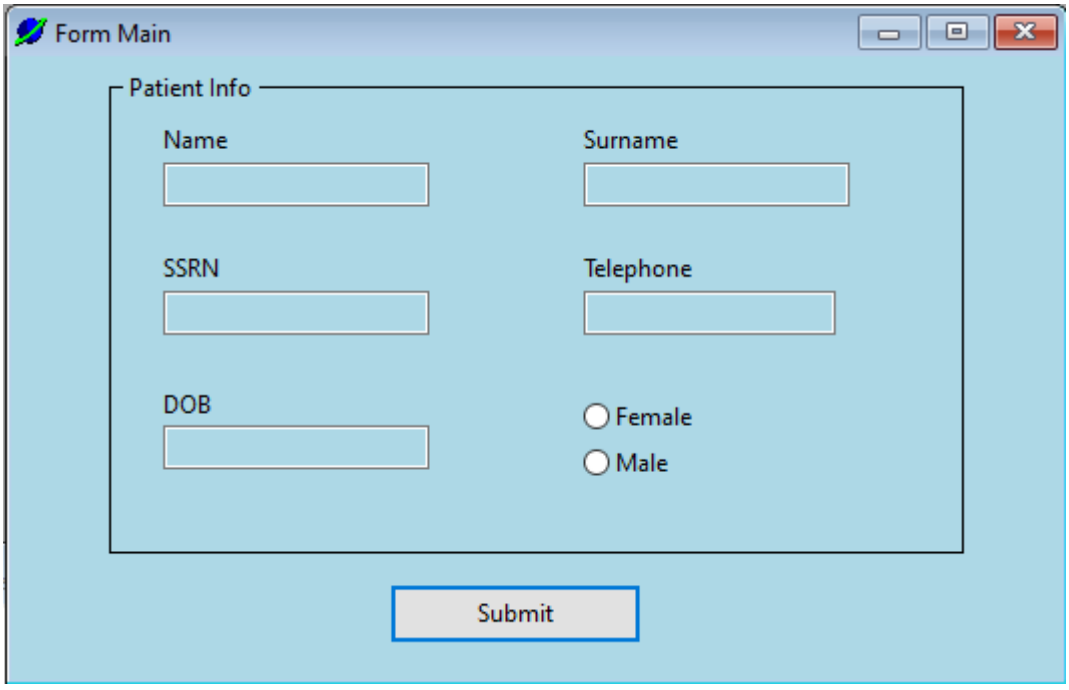
ΕΙΚΟΝΑ 38 : ΔΕΥΤΕΡΟ ΒΗΜΑ ΕΚΤΕΛΕΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Ολοκληρώνοντας τα παραπάνω βήματα, θα ανοίξει στον χρήστη ένα παράθυρο καλωσορίσματος (Welcome Form), ώστε να εισάγει τα στοιχεία του. Κάνοντας κλικ στο κουμπι ‘’Proceed’’.



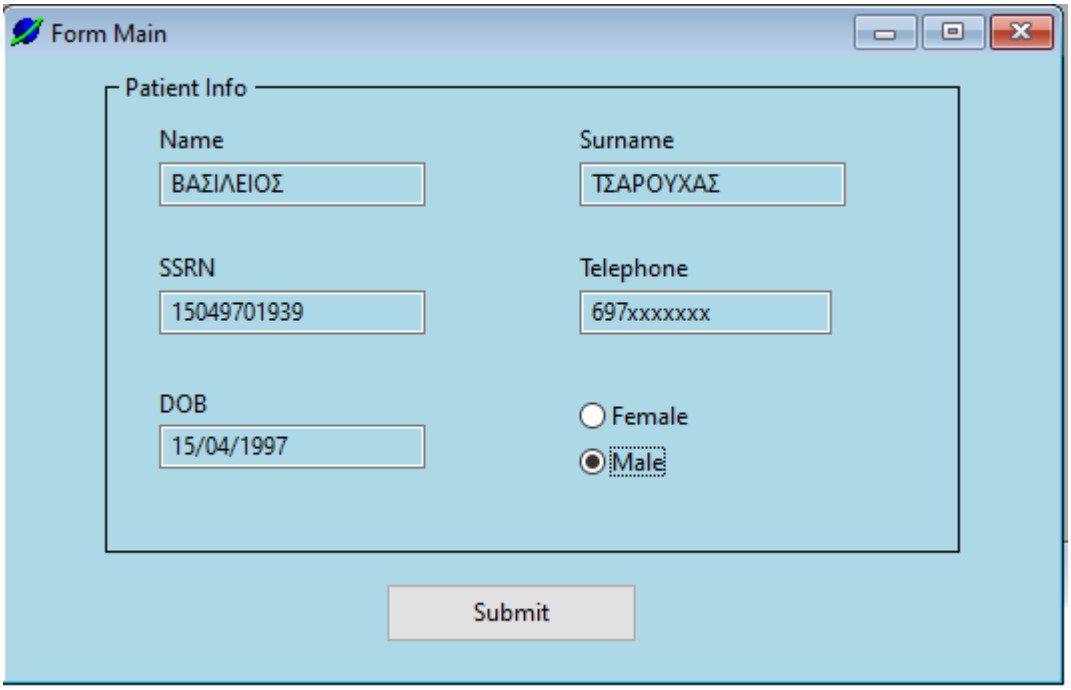
ΕΙΚΟΝΑ 39 : ΦΟΡΜΑ ΚΑΛΩΣΟΡΙΣΜΑΤΟΣ

Πατώντας το κουμπί ανοίγει η παρακάτω φόρμα συμπλήρωσης στοιχείων χρήστη – ασθενή.



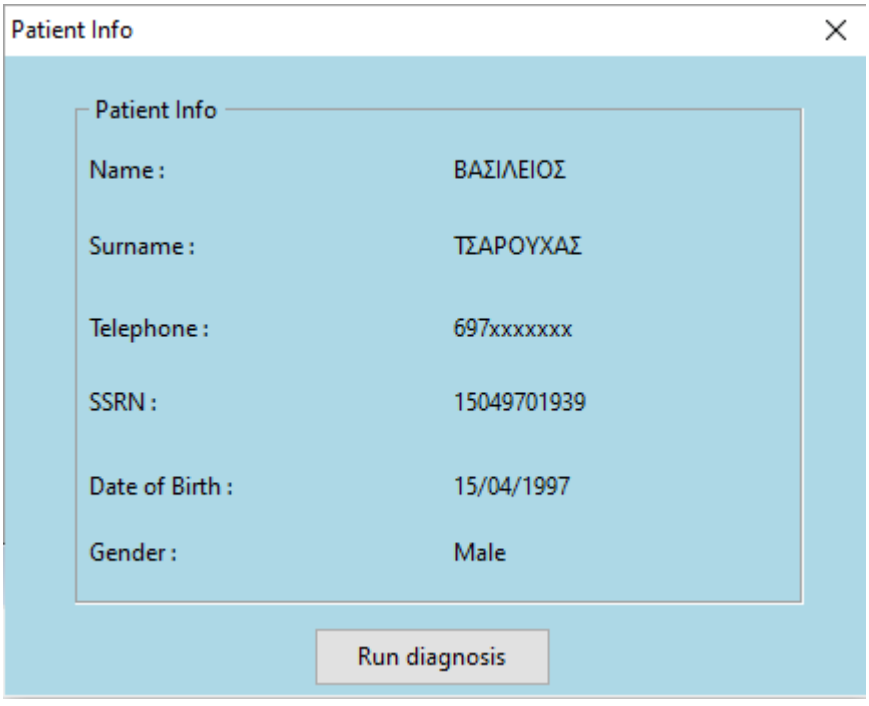
ΕΙΚΟΝΑ 40 : ΦΟΡΜΑ ΣΥΜΛΗΡΩΣΗΣ ΣΤΟΙΧΕΙΩΝ

Στη συνέχεια, συμπληρώνοντας όλα τα στοιχεία του ο χρήστης, στην Form Main. Για να ξεκινήσει η διάγνωση θα πρέπει να πατήσει το κουμπί ‘’Submit’’.



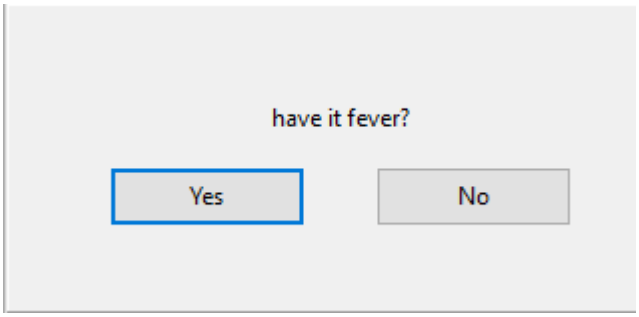
ΕΙΚΟΝΑ 41 : ΣΥΜΠΛΗΡΩΜΕΝΗ ΦΟΡΜΑ ΣΤΟΙΧΕΙΩΝ

Κάνοντας κλικ στο κουμπί, θα ανοίξει η φόρμα Patient Info. Η οποία θα ανακεφαλαιώνει όλα τα στοιχεία που καταχώρησε στο προηγούμενο βήμα ο χρήστης.



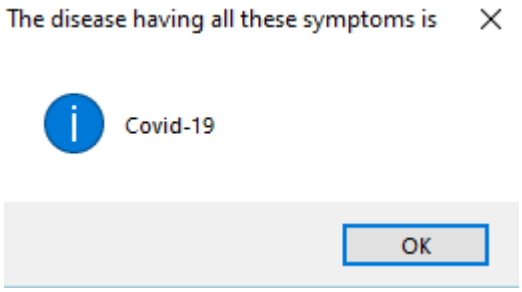
ΕΙΚΟΝΑ 42 : ΦΟΡΜΑ ΕΞΑΚΡΙΒΩΣΗΣ ΣΤΟΙΧΕΙΩΝ

Για να ξεκινήσουν οι ερωτήσεις, για την διάγνωση της ασθένειας του χρήστη. Θα πρέπει να πατήσει το κουμπί ‘’Run diagnosis’’. Αφού, πατήσει το κουμπί, θα εμφανιστεί ένα παράθυρο διαλόγου.



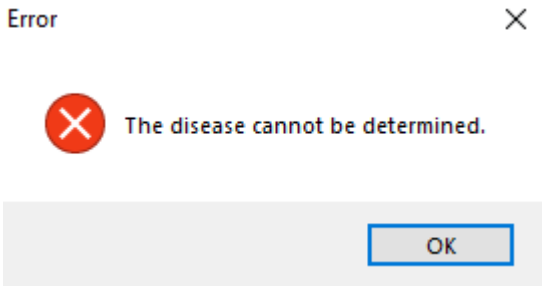
ΕΙΚΟΝΑ 43 : ΠΑΡΑΘΥΡΟ ΔΙΑΛΟΓΟΥ

Αυτό το παράθυρο διαλόγου αποτελείται από μία ερώτηση σχετική με κάποιο σύμπτωμα ασθένειας. Αν ο χρήστης πατήσει Yes (ναι), σε όλους του κανόνες που αφορούν μία ασθένεια. Τότε το σύστημα θα εμφανίσει παράθυρο μηνύματος με την διάγνωση της ασθένειας του χρήστη.



ΕΙΚΟΝΑ 44 : ΜΗΝΥΜΑ ΔΙΑΓΝΩΣΗΣ

Διαφορετικά, αν ο χρήστης απαντήσει στους περισσότερους κανόνες με No (όχι). Τότε το σύστημα δε θα μπορέσει να κάνει κάποια σωστή διάγνωση και θα εμφανίσει μήνυμα σφάλματος.

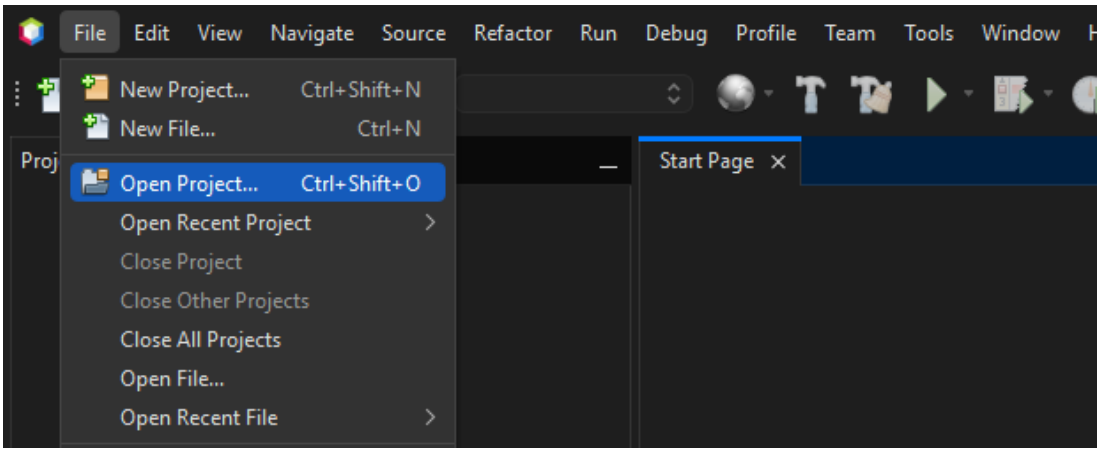


ΕΙΚΟΝΑ 45 : ΜΗΝΥΜΑ ΣΦΑΛΜΑΤΟΣ

3.4 Παρουσίαση εφαρμογής Java

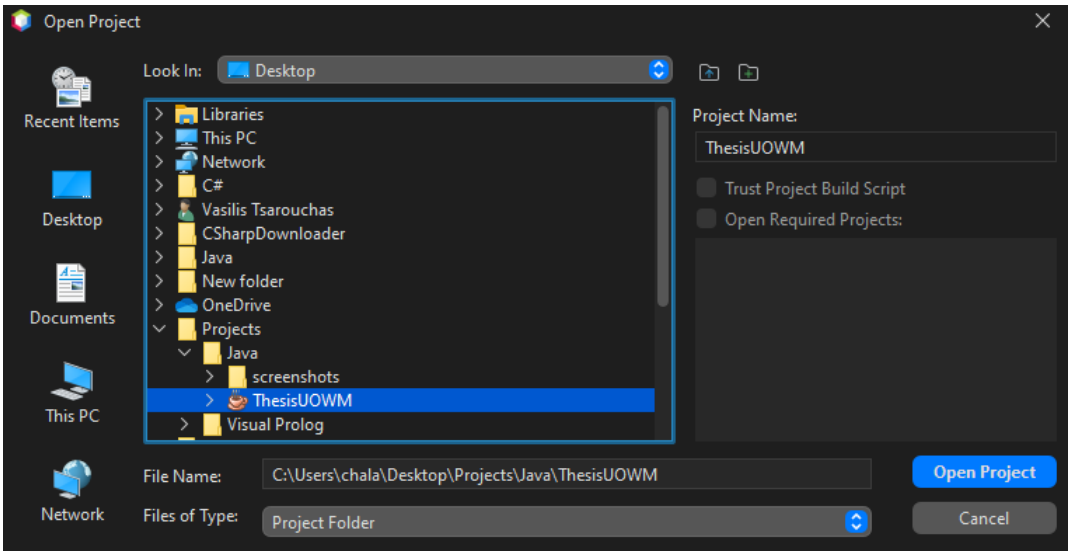
Σε αυτή την ενότητα θα γίνει η παρουσίαση της εφαρμογής, με τα βήματα που θα πρέπει να ακολουθήσει ο χρήστης, στο γραφικό περιβάλλον της εφαρμογής στη Java.

Αρχικά ο χρήστης θα πρέπει να έχει εγκατεστημένο το Netbeans στον προσωπικό του υπολογιστή, μαζί με τις απαραίτητες βιβλιοθήκες. Αφού γίνει η εγκατάσταση, ο χρήστης θα πρέπει να ανοίξει το πρόγραμμα, με τα ακόλουθα βήματα File -> Open Project.



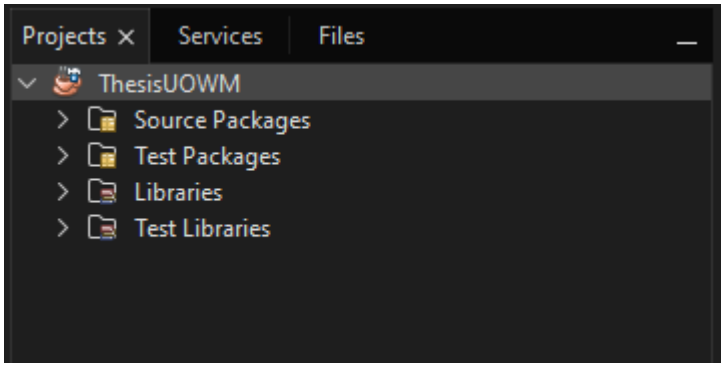
ΕΙΚΟΝΑ 46 : ΠΡΩΤΟ ΒΗΜΑ ΕΙΣΑΓΩΓΗΣ ΑΡΧΕΙΟΥ- ΕΦΑΡΜΟΓΗΣ

Στη συνέχεια θα αναζητήση και θα επιλέξει τον φάκελο της εφαρμογής.



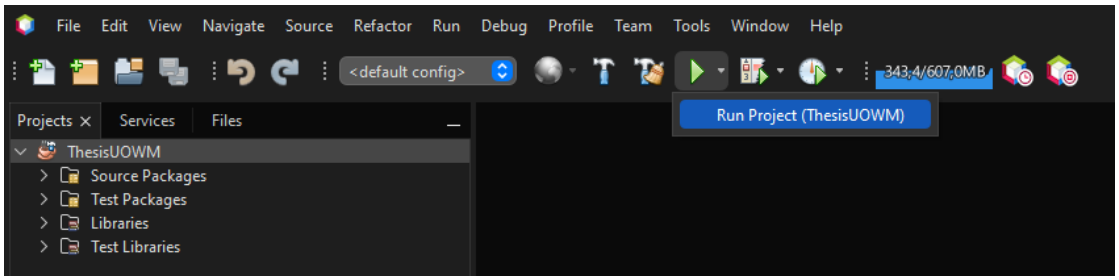
ΕΙΚΟΝΑ 47 : ΕΠΙΛΟΓΗ ΑΡΧΕΙΟΥ-ΕΦΑΡΜΟΓΗΣ

Κάνοντας κλικ στο κουμπί ‘’Open Project’’, θα φορτωθεί το πρόγραμμα στο περιβάλλον ανάπτυξης. Το οποίο θα μας επιτρέψει την εκτέλεση του.



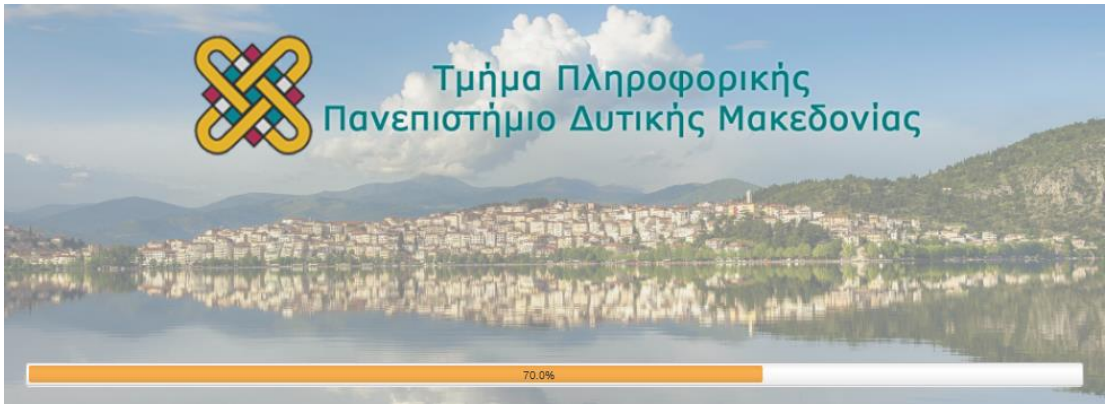
ΕΙΚΟΝΑ 48 : ΟΛΟΚΛΗΡΩΣΗ ΕΙΣΑΓΩΓΗΣ

Εφόσον τα παραπάνω βήματα εκτελεστούν σωστά από τον χρήστη, θα πρέπει στην καρτέλα Projects να εμφανίζεται το πρόγραμμα, μαζί με τα απαραίτητα αρχεία του.



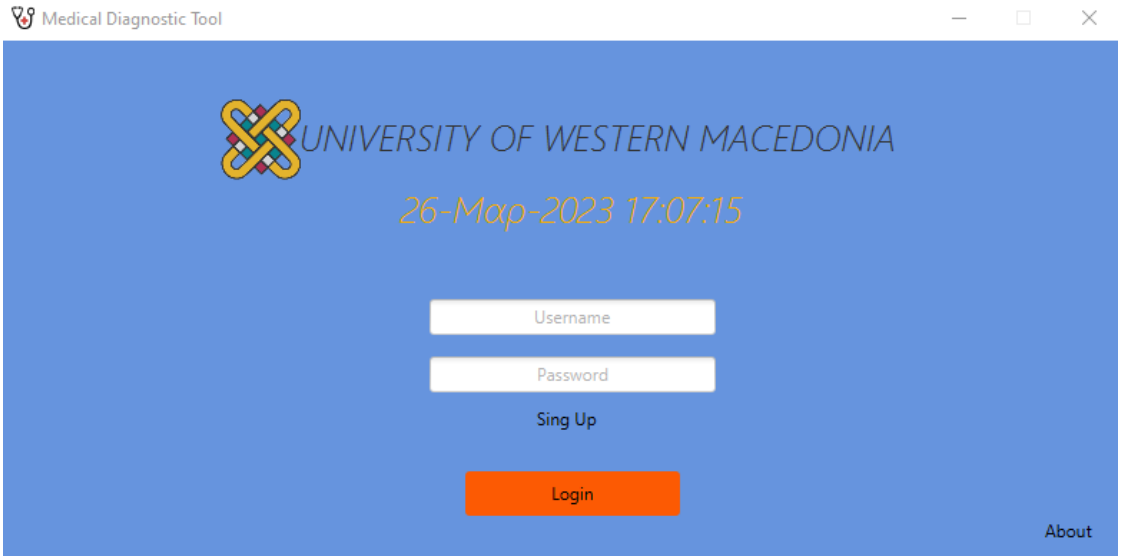
ΕΙΚΟΝΑ 49 : ΕΚΤΕΛΕΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Έπειτα ο χρήστης για να εκτελέσει το πρόγραμμα θα πρέπει από την μπάρα εργαλείων να επιλέξει το κουμπί της εκτέλεσης όπως αυτό φαίνεται στην παραπάνω εικόνα.



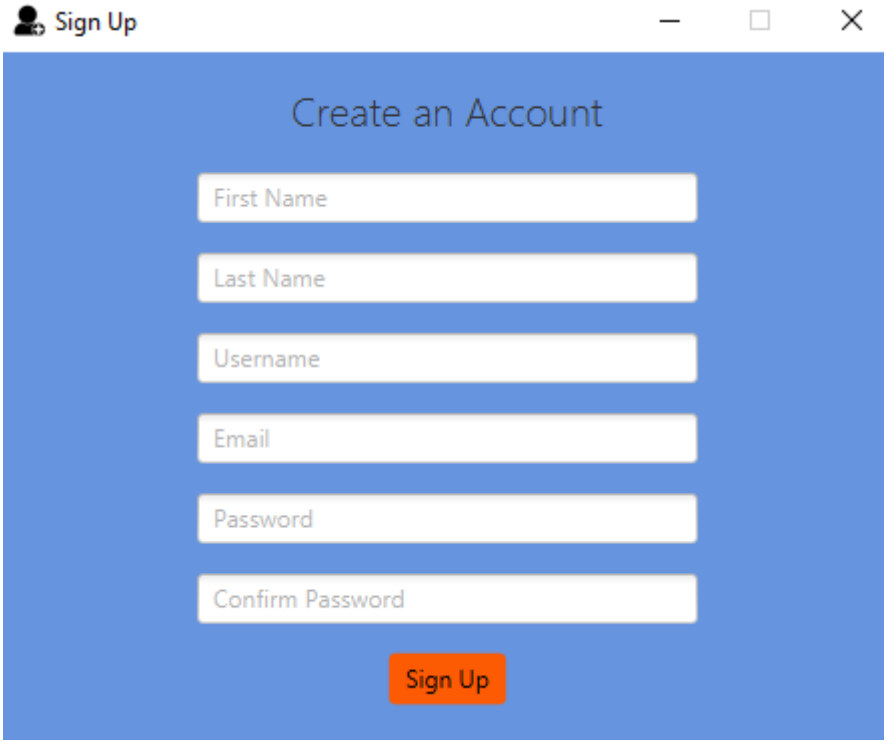
ΕΙΚΟΝΑ 50 : ΟΘΟΝΗ ΕΚΚΙΝΗΣΗΣ

Στη συνέχεια εμφανίζεται η οθόνη εκκίνησης. Όταν γεμίσει πλήρως η μπάρα, εμφανίζεται η φόρμα σύνδεσης.



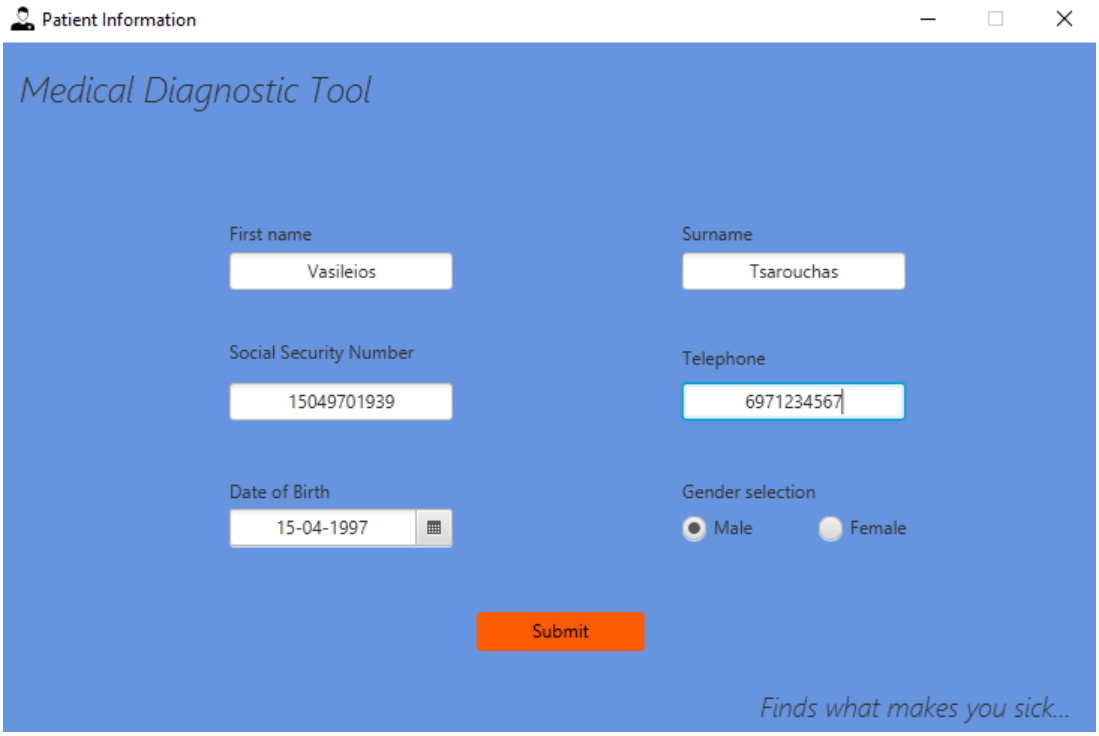
ΕΙΚΟΝΑ 51 : ΦΟΡΜΑ ΣΥΝΔΕΣΗΣ ΧΡΗΣΤΗ

Στη φόρμα σύνδεσης ο χρήστης καλείται, είτε να συνδεθεί με τα προσωπικά του διαπιστευτήρια, είτε να κάνει εγγραφή.



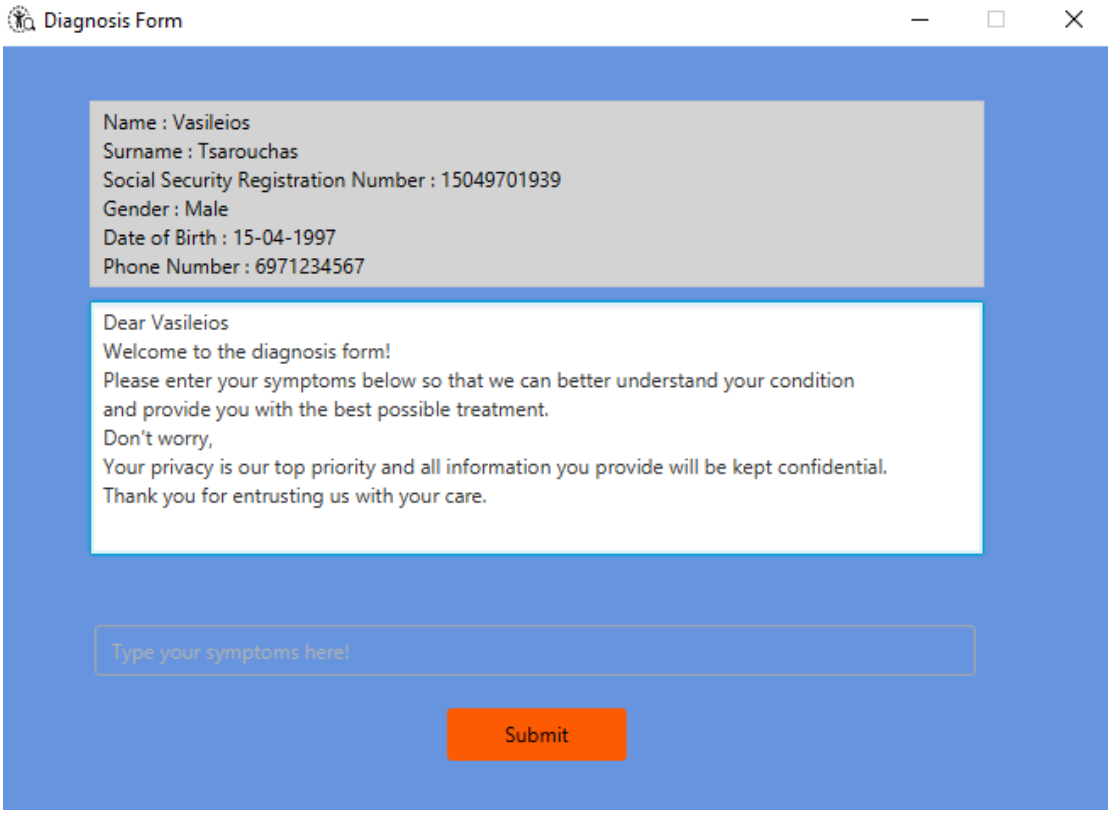
ΕΙΚΟΝΑ 52 : ΦΟΡΜΑ ΕΓΓΡΑΦΗΣ ΧΡΗΣΤΗ

Αφού ολοκληρωθεί η εγγραφή του χρήστη. Ο χρήστης ξανά μεταφέρεται στην φόρμα σύνδεσης, για να εισάγει τα διαπιστευτήρια του και να πραγματοποιήσει σύνδεση.



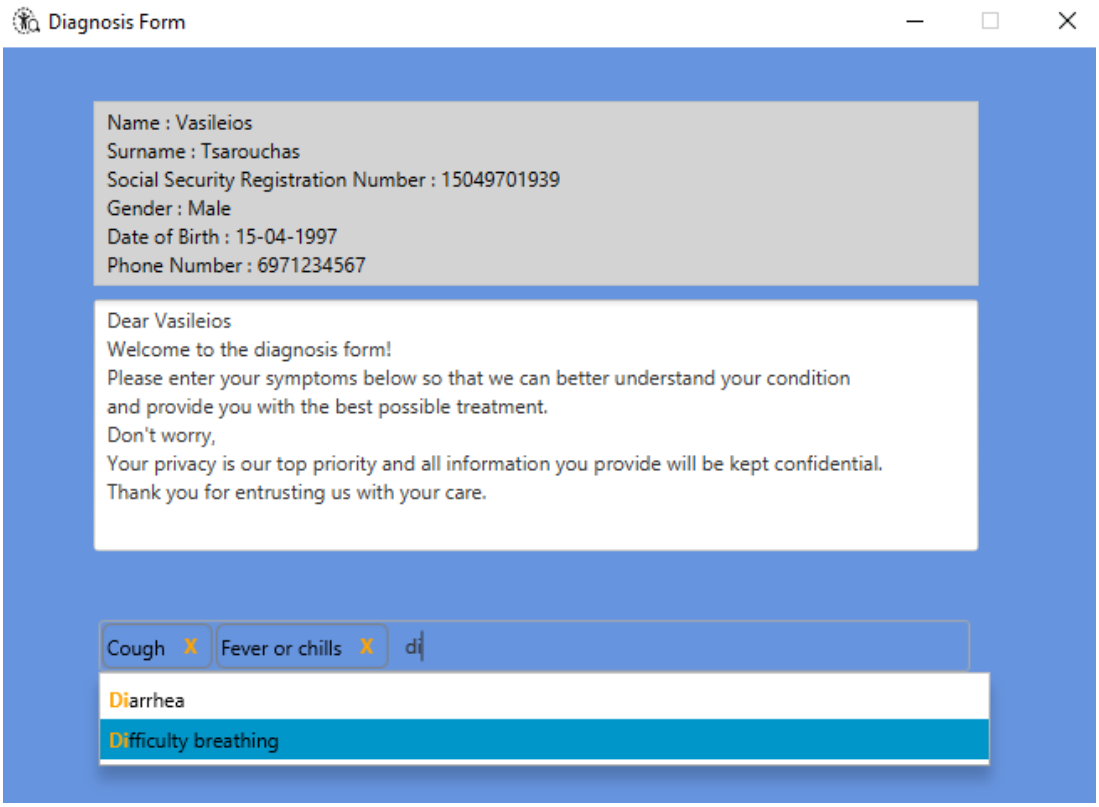
ΕΙΚΟΝΑ 53 : ΦΟΡΜΑ ΔΗΛΩΣΗΣ ΣΤΟΙΧΕΙΩΝ

Στη συνέχεια εμφανίζεται η φόρμα Patient Information. Αν έχει χρησιμοποιήσει την εφαρμογή στο παρελθόν η καρτέλα εμφανίζεται συμπληρωμένη. Διαφορετικά αν είναι η πρώτη φορά που κάνει εγγραφή και σύνδεση, θα πρέπει να συμπληρώσει τα υπόλοιπα πεδία της φόρμας. Εκτός απο το ονοματεπώνυμο που εκχωρείται από την βάση δεδομένων, από την προηγούμενη καρτέλα που έκανε εγγραφή. Κάνοντας κλικ στο κουμπί ‘’Submit’’ αποθηκεύονται τα στοιχεία του στη βάση δεδομένων.



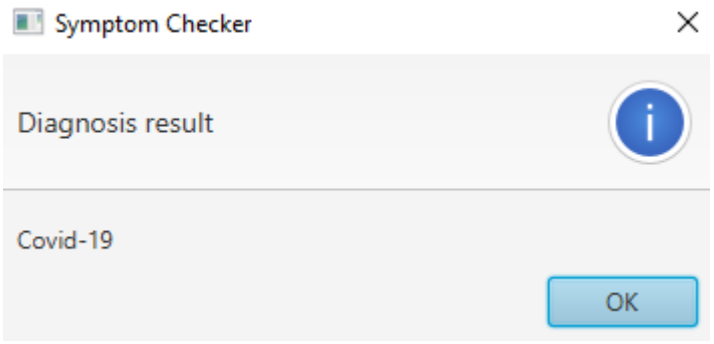
ΕΙΚΟΝΑ 54 : ΦΟΡΜΑ ΔΗΛΩΣΗΣ ΣΥΜΠΤΩΜΑΤΩΝ

Έπειτα εμφανίζεται το παράθυρο Diagnosis Form, το οποίο εμφανίζει τα στοιχεία του χρήστη και ένα μήνυμα καλωσορίσματος. Στο πεδίο κειμένου ο χρήστης έχει την δυνατότητα να εισάγει τα συμπτώματα του.



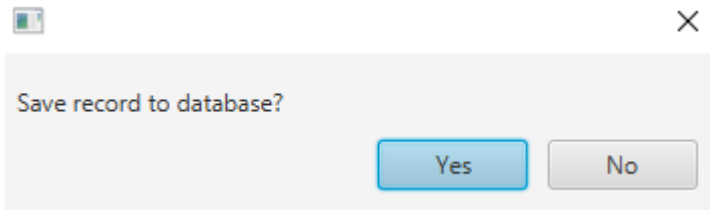
ΕΙΚΟΝΑ 55 : ΦΟΡΜΑ ΚΑΤΑΧΩΡΗΣΗΣ ΣΥΜΠΤΩΜΑΤΩΝ

Κάνοντας κλικ στο κουμπί ‘’Submit’’ ξεκινάει η διαδικασία δίαγνωσης.



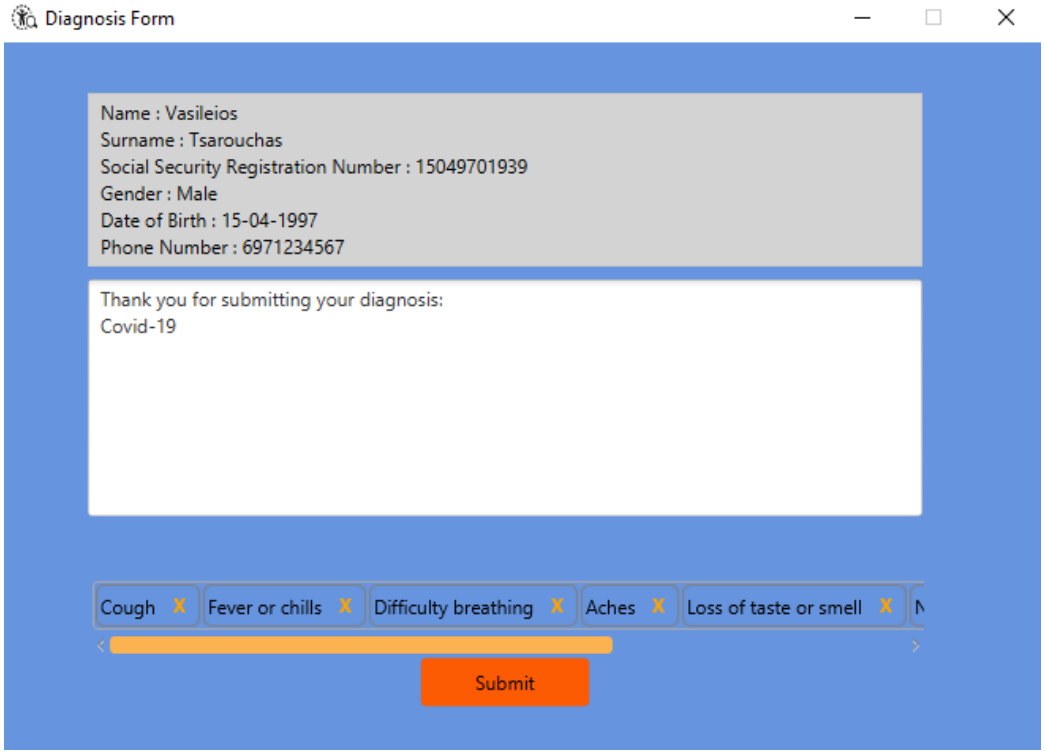
ΕΙΚΟΝΑ 56 : ΑΠΟΤΕΛΕΣΜΑ ΔΙΑΓΝΩΣΗΣ

Αν βρεθεί διάγνωση εμφανίζεται μήνυμα με το αποτέλεσμα της ασθένειας.



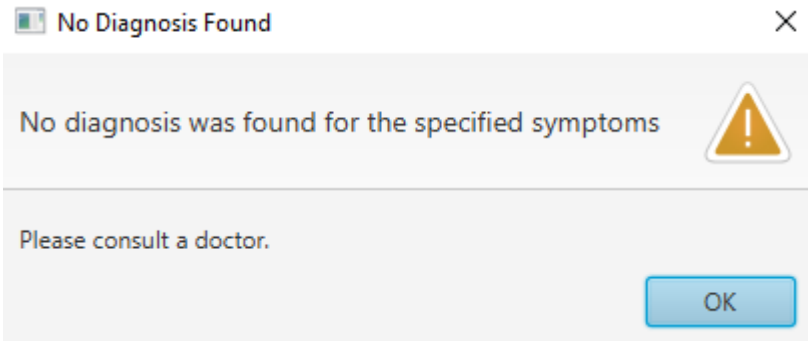
ΕΙΚΟΝΑ 57 : ΕΡΩΤΗΣΗ ΓΙΑ ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Αφότου γίνει η διάγνωση γίνεται ερώτηση στον χρήστη, εάν επιθυμεί την αποθήκευση του αποτελέσματος στην βάση δεδομένων.



ΕΙΚΟΝΑ 58 : ΕΠΙΤΥΧΗΣ ΚΑΤΑΧΩΡΗΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Τότε στο πεδίο κειμένου εμφανίζεται ότι καταχωρήθηκε η διάγνωση στη βάση δεδομένων και το όνομα της ασθένειας του χρήστη.



ΕΙΚΟΝΑ 59 : ΜΗΝΥΜΑ ΣΦΑΛΜΑΤΟΣ

Στην περίπτωση που δεν βρεθεί διάγνωση, εμφανίζεται στον χρήστη το παραπάνω μήνυμα.

3.5 Συγκριτική ανάλυση

Η Java και η Visual Prolog είναι δύο πολύ διαφορετικές γλώσσες προγραμματισμού με ξεχωριστά πλεονεκτήματα και μειονεκτήματα. Ακολουθεί μια λεπτομερής σύγκριση των δύο γλωσσών.

Τόσο η Java όσο και η Visual Prolog είναι αντικειμενοστραφείς γλώσσες προγραμματισμού. Ωστόσο, η Java είναι μια πιο ευρέως χρησιμοποιούμενη αντικειμενοστραφής γλώσσα με ένα τεράστιο και καθιερωμένο οικοσύστημα εργαλείων και βιβλιοθηκών για την κατασκευή σύνθετων συστημάτων λογισμικού.

Η σύνταξη της Java είναι πιο διαισθητική και πιο κοντά στις C/C++ και C#. Η σύνταξη της Visual Prolog βασίζεται στην Prolog, η οποία είναι μια γλώσσα λογικού προγραμματισμού, και μπορεί να χρειαστεί κάποια εξοικείωση. Ειδικά για τους προγραμματιστές που είναι περισσότερο, εξοικειωμένοι με προστακτικές ή αντικειμενοστραφείς γλώσσες προγραμματισμού.

Η Java είναι γνωστή για την αρχή της "Write Once, Run Anywhere", η οποία επιτρέπει στα προγράμματα Java να εκτελούνται σε οποιονδήποτε υπολογιστή ή συσκευή με εγκατεστημένο JVM. Αντίθετα, η Visual Prolog έχει σχεδιαστεί κυρίως για λειτουργικά συστήματα Windows και δεν έχει το ίδιο επίπεδο ανεξαρτησίας πλατφόρμας με τη Java.

Η Java διαθέτει μια τεράστια τυπική βιβλιοθήκη με εκτεταμένες λειτουργίες για τα πάντα, από τη δικτύωση έως την επεξεργασία XML. Διαθέτει επίσης μια τεράστια και καθιερωμένη κοινότητα βιβλιοθηκών και πλαισίων ανοικτού κώδικα που μπορούν να χρησιμοποιηθούν για μία ποικιλία εργασιών. Η Visual Prolog διαθέτει μία μικρότερη τυπική βιβλιοθήκη που επικεντρώνεται κυρίως στον προγραμματισμό GUI.

Η Java διαθέτει αυτόματη διαχείριση μνήμης μέσω της συλλογής σκουπιδιών, η οποία μπορεί να μειώσει τις διαρροές μνήμης και να απλοποιήσει τη διαχείριση μνήμης για τους προγραμματιστές. Η Visual Prolog έχει το δικό της σύστημα διαχείρισης μνήμης που βασίζεται στην καταμέτρηση αναφορών.

Η Java θεωρείται γενικά ότι είναι μια γρήγορη γλώσσα, εν μέρει χάρη στον μεταγλωττιστή Just-in-Time (JIT), ο οποίος μεταφράζει τον bytecode σε εγγενή κώδικα μηχανής. Η Visual Prolog είναι πιο αργή από τη Java, καθώς είναι διερμηνευμένη και όχι μεταγλωττισμένη.

Η Java χρησιμοποιείται ευρέως για μία ποικιλία εφαρμογών, συμπεριλαμβανομένων των εφαρμογών ιστού, των εφαρμογών για κινητά τηλέφωνα και του λογισμικού γραφείου. Η Visual Prolog χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών βασισμένων σε κανόνες και έμπειρα συστήματα.

Συνολικά, η Java και η Visual Prolog είναι δύο πολύ διαφορετικές γλώσσες προγραμματισμού με διαφορετικές περιπτώσεις χρήσης. Ενώ η Java είναι μία πιο ευρέως χρησιμοποιούμενη και ευέλικτη γλώσσα. Η Visual Prolog έχει μοναδικά πλεονεκτήματα για την ανάπτυξη εφαρμογών βασισμένων σε κανόνες και έμπειρα συστήματα. Οι προγραμματιστές θα πρέπει να εξετάζουν τις ειδικές ανάγκες του έργου τους όταν επιλέγουν μεταξύ των δύο γλωσσών.

Οι διαφορές που έχουν αυτές οι δύο γλώσσες προγραμματισμού, έγιναν γρήγορα αντιληπτές κατά την υλοποίηση αλλά και την σχεδίαση της εφαρμογής. Όμως υπήρχαν και μερικά σημεία κατά τη διάρκεια της υλοποίησης που το αποτέλεσμα ήταν αν όχι ίδιο σχεδόν παρομοιότυπο στις δύο γλώσσες. Παρακάτω παρατίθενται οι δικές μας απόψεις και εκτιμήσεις, σχετικά με την σύγκριση τους, στον τομέα αποκλειστικά της δημιουργίας και του σχεδιασμού της εφαρμογής.

Αφενός η Java έχει πολλούς υποστηρικτές, άρα έχει μεγάλη ποικιλία σε παραδείγματα και βιβλιογραφία. Αυτό μας διευκόλυνε να αντλήσουμε έναν ευρύ αριθμό παραδειγμάτων και οδηγιών, από το διαδίκτυο, για την υλοποίηση της εφαρμογής. Επίσης υπάρχει και αρκετό υλικό για την JavaFX και όλα όσα προσφέρει. Υπάρχει η δυνατότητα τροποποίσης του γραφικού περιβάλλοντος, με μόνο περιορισμό την φαντασία του προγραμματιστή. Η χρήση βάσης δεδομένων στην Java υλοποιήθηκε χωρίς ιδιαίτερες δυσκολίες. Για την κατασκευή της εφαρμογή χρησιμοποιήσαμε το εργαλειο Netbeans, για το οποίο υπάρχει μεγάλη ποικιλία υλικού, τόσο για την εγκατάσταση του προγράμματος, όσο και για τον χειρισμό του.

Αφετέρου η Visual Prolog έχει μικρή κοινότητα χρηστών, άρα έχει και περιορισμένη ποικιλία σε παραδείγματα και βιβλιογραφία. Αυτό μας δυσκόλεψε αρκετά στο να αντλήσουμε πληροφορίες, παραδείγματα και οδηγίες. Τόσο για την σύνταξη του κώδικα όσο και για την χρήση των αντικειμένων του γραφικού περιβάλλοντος. Επιπλέον υπάρχει περιορισμός στο περιβάλλον ανάπτυξης. Διότι, η Visual Prolog έχει μόνο ένα περιβάλλον ανάπτυξης, το οποίο αφορά μόνο την ίδια γλώσσα και καμία άλλη.

Τέλος να σημειωθεί ότι στην εφαρμογή της Visual Prolog ο χρήστης απαντάει σε ερωτήσεις με Ναι ή Οχι. Οι οποίες είναι σχετικές με τα συμπτώματα, των ασθενειών που είναι καταχωρημένες στη βάση δεδομένων. Ενώ στην εφαρμογή της Java ο χρήστης καλείται να συμπληρώσει στο πεδίο κειμένου, μόνος του τα συμπτώματα της ασθένειας που έχει. Ο λόγος που χρησιμοποιήθηκε διαφορετική μέθοδος για την υλοποίηση της εφαρμογής σε αυτές τις δύο γλώσσες. Ήταν για να δείξει τη βασική διαφορά που έχουν μεταξύ τους και είναι ο χρόνος της διάγνωσης.Όπως επίσης και η έκταση του κώδικα, για την υλοποίηση ενός πολύπλοκου κανόνα-συστήματος, είναι πολύ μικρότερη στην Visual Prolog σε συγκριση με την Java.

Ωστόσο, υπάρχουν κάποια κοινά μεταξύ τους στη δήλωση των μεταβλητών, των σταθερών, των κλάσεων και των τρόπων αλληλεπίδρασης των στοιχείων που περιέχουν ένα γεγονός ενεργοποίησης. Για παράδειγμα κάνοντας κλικ σε κάποιο κουμπί ανοίγει ένα παράθυρο ή εμφανίζεται κάποιο μήνυμα στην οθόνη.

Συμπεράσματα

Σκοπός της παρούσης πτυχιακής εργασίας ήταν ο σχεδιασμός – δημιουργία μίας ιατρικής, διαγνωστικής εφαρμογής. Με την χρήση δύο αρκετά διαφορετικών μεταξύ τους γλωσσών προγραμματισμού, την Visual Prolog και την Java. Η δημιουργία μίας διαγνωστικής εφαρμογής είναι δύσκολη. Διότι απαιτεί από τον προγραμματιστή να συλέξει πρώτα όλα τα στοιχεία που χρειάζεται για την υλοποιήση της εφαρμογής. Να τα κατανοήσει και έπειτα να τα ταξινομήσει κατά βήματα, ώστε να μπορέσει μετά να τα εισάγει στο πρόγραμμα πιο εύκολα.

Η υλοποίηση της εφαρμογής ήταν μία πρόκληση για εμας. Γιατί απαιτούσε γνώσεις αρχιτεκτονικής και σχεδίασης. Έπρεπε να μάθουμε τόσο την γλώσσα προγραμματισμού Visual Prolog, όσο και το γραφικό της περιβάλλον. Με την γλώσσα προγραμματισμού Java ήμασταν πιο εξοικειωμένοι αλλά και εκεί χρειαστήκαμε περαιτέρω εξέλιξη στις γνώσεις μας. Ωστόσο, αξίζει να σημειωθεί ότι καταβάλαμε κάθε δυνατή προσπάθεια, για να είναι η εφαρμογή πλήρως λειτουργική.

Η εφαρμογή εκπληρώνει τον σκοπό της αλλά εξακολουθεί να μπορεί να βελτιωθεί περαιτέρω. Για παράδειγμα ως μελλοντική εξέλιξη της εφαρμογής και στις δύο γλώσσες προγραμματισμού (Java και Visual Prolog) μπορούν να προστεθούν και άλλες ασθένειες, μαζί με τα συμπτώματα τους. Πιο αναλυτικά στην εφαρμογή της Java, θα μπορούσε να μεταφερθεί και σε κινητές συσκευές. Να ενσωματωθεί ένα API, ώστε αυτό να είναι υπεύθυνο για την επικοινωνία της βάσης δεδομένων. Επιπλέον θα μπορούσε η εφαρμογή σε περίπτωση που ο χρήστης – ασθενής δεν καταχωρήσει σωστά όλα τα συμπτώματα κάποιας ασθένειας, να γίνεται έλεγχος σε όλες τις ασθένειες και όποιες περιέχουν τα περισσότερα κριτήρια (τα κρητίρια είναι τα συμπτώματα που έχει καταχωρήσει ο χρήστης) να εμφανίζει τις εκάστοτε ασθένειες ποσοστιαία και με κατάλληλο μήνυμα εξήγησης. Τέλος για την Visual Prolog θα μπορούσε να μη χρησιμοποιηθεί η ήδη ενσωματωμένη βάση δεδομένων που έχει. Αλλά να γίνει χρήση μίας εξωτερικής βάσης (τύπου SQLite), για να υπάρχει και εκεί η δυνατότητα δημιουργίας - σύνδεσης χρήστη και η αποθήκευση αποτελεσμάτων της διάγνωσης.

Βιβλιογραφία

[1] Wikipedia. Diagnostic program. 15/11/2022 [online] https://en.wikipedia.org/wiki/Diagnostic_program [Accessed 19/03/2022]

[2] Wikipedia. Clinical decision support system. 25/03/2023 [online] https://en.wikipedia.org/wiki/Clinical_decision_support_system [Accessed 19/03/2023]

[3] Centers for Disease Control and Prevention (CDC). 26/10/2022. Symptoms of COVID-19. [online] https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html [Accessed 10/02/2023]

[4] Centers for Disease Control and Prevention (CDC). 27/06/2022 Group A Streptococcal (GAS) Disease. [online] https://www.cdc.gov/groupastrep/diseases-hcp/strep-throat.html [Accessed 10/02/2023]

[5],[6] Centers for Disease Control and Prevention (CDC). 18/05/2022 Symptoms and Complications. [online] https://www.cdc.gov/pneumococcal/about/symptoms-complications.html [Accessed 10/02/2023]

[7] Centers for Disease Control and Prevention (CDC). 29/09/2022 Cold Versus Flu. [online] https://www.cdc.gov/flu/symptoms/coldflu.htm [Accessed 11/02/2023]

[8] Centers for Disease Control and Prevention (CDC). 18/11/2021 Flu Symptoms & Diagnosis. [online] https://www.cdc.gov/flu/symptoms/ [Accessed 11/02/2023]

[9] Thomas W. De Boer. A Beginners' Guide to Visual Prolog version 7.2. Groningen, Academia, 2009. https://www.academia.edu/8999914/A_Beginners_Guide_to_Visual_Prolog

[10] Jimmy Singla, Dinesh Grover and Abhinav Bhandari. Medical Expert Systems for Diagnosis of Various Diseases. International Journal of Computer Applications (0975-8887) volume 93 No.7 May 2014. [online] https://www.researchgate.net/profile/Jimmy-Singla/publication/271157162_Medical_Expert_Systems_for_Diagnosis_of_Various_Diseases/links/5ad191daaca272fdaf779ffa/Medical-Expert-Systems-for-Diagnosis-of-Various-Diseases.pdf

[11] Savitch Walter. JAVA μια εισαγωγή στην επίλυση προβλημάτων και στον προγραμματισμό, 7η έκδοση. Εκδόσεις Α. ΤΖΙΟΛΑ & YIOI Α.Ε, 2016.

[12] Wikipedia . WebMD. 28/03/2023 [online] https://en.wikipedia.org/wiki/WebMD [Accessed 23/03/2023]

[13] Ada Health GmbH. Ada. 2023 [online] https://ada.com/ [Accessed 20/03/2023]

[14]Wikipedia. VisualDx. 28/01/2020 [online] https://en.wikipedia.org/wiki/VisualDx [Accessed 20/03/2023]

[15] Randall Scott. a guide to ARTIFICIAL INTELLIGENCE with VISUAL PROLOG. Denver, Outskirts Press Inc, 2010.

[16] Geertjan Wielenga. Beginning NetBeans IDE For Java Developers. Berkeley, Apress, 2015. [online] Available: https://books.google.gr/books?hl=el&lr=&id=9HeBCgAAQBAJ&oi=fnd&pg=PR6&dq=netbeans+ide+&ots=9EWrPqqCON&sig=dKax9Ncgiujlr6O57805BWE01QI&redir_esc=y#v=onepage&q=netbeans%20ide&f=false

[17] Kim Topley. JavaFX Developer's Guide. Boston, Addison-Wesley, 2010. [omline] Available: https://books.google.gr/books?hl=el&lr=&id=bnDCXcoZR9MC&oi=fnd&pg=PT34&dq=javafx&ots=hb3VNFJKie&sig=BQPDMirz526TKjmX7AIpg0Z7zYU&redir_esc=y#v=onepage&q=javafx&f=false

[18] Grant Allen and Mike Owens. The Definitive Guide to SQLite (second edition). Berkeley, Apress 2010. [onlie] Available: https://www.programmershouse.ir/Library/Files/SQLite.pdf

[19] Yongzhen Ou. On Mapping between UML and Entity-Relationship Model. Σχολή Μαθηματικών και Επιστήμης Υπολογιστών, University of Konstanz, Germany, 1998. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ddde3f2f9b972b371222014d0ff6777ce0ba6d1d

[20] Ahmad Al-ShanailH. An Experimental Comparison of ER and UML Class Diagrams. Mutah University, Jordan, 2015. https://gvpress.com/journals/IJHIT/vol8_no2/26.pdf

[21] Vili Podgorelec, Peter Kokol, Bruno Stiglic and Ivan Rozman. Decision trees: an overview and their use in medicine. University of Maribor, Slovenia, 2002. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0af1219b2af972e362ed879f4386338b444e6a02

[22] Oscar Cordón, Francisco Herrera, and Pedro Villar. Generating the Knowledge Base of a Fuzzy Rule-Based System by the Genetic Learning of the Data Base. University of Granada, Spain, 2001. https://www.researchgate.net/publication/3336006_Generating_the_knowledge_base_of_a_fuzzy_rule-based_system_by_the_genetic_learning_of_the_data_base

[23] Samy S. Abu Naser and Bashar G. Bastami. A Proposed Rule Based System for Breasts Cancer Diagnosis. WWJMRD. Faculty of Engineering & Information Technology, Al-Azhar University, Gaza, Palestine, 2016. http://dstore.alazhar.edu.ps/xmlui/bitstream/handle/123456789/297/A_Proposed_Rule_Based_System_for_Breasts_Cancer_Di.pdf?sequence=1&isAllowed=y

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

Visual Prolog

```
implement main

clauses
    run() :-
        TaskWindow = taskWindow::new(),
        TaskWindow:show().

end implement main

goal
    mainExe::run(main::run).


% Copyright Tsarouchas Vasileios

implement taskWindow inherits applicationWindow
    open core, vpiDomains

constants
    mdiProperty : boolean = true.

clauses
    new() :-
        applicationWindow::new(),
        generatedInitialize().

predicates
    onShow : window::showListener.
clauses
    onShow(_, _CreationData) :-
        _MessageForm = messageForm::display(This).

class predicates
    onDestroy : window::destroyListener.
clauses
    onDestroy(_).

class predicates
    onHelpAbout : window::menuItemListener.
clauses
    onHelpAbout(TaskWin, _MenuTag) :-
```

```
        _AboutDialog = aboutDialog::display(TaskWin).

predicates
    onFileExit : window::menuItemListener.
clauses
    onFileExit(_, _MenuTag) :-
        close().

predicates
    onSizeChanged : window::sizeListener.
clauses
    onSizeChanged(_) :-
        vpiToolbar::resize(getVPIWindow()).

predicates
    onFileNew : window::menuItemListener.
clauses
    onFileNew(_Source, _MenuTag) :-
        _ = preMain::display(This).

% This code is maintained automatically, do not update it manually.
predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("ThesisUOWM"),
        setDecoration(titlebar([frameDecoration::closeButton, frameDecoration::m
aximizeButton, frameDecoration::minimizeButton])),
        setBorder(frameDecoration::sizeBorder),
        setState([wsf_ClipSiblings]),
        whenCreated({  :- projectToolbar::create(getVpiWindow()) }),
        setMdiProperty(mdiProperty),
        menuSet(resMenu(resourceIdentifiers::id_TaskMenu)),
        addShowListener(onShow),
        addSizeListener(onSizeChanged),
        addDestroyListener(onDestroy),
        addMenuItemListener(resourceIdentifiers::id_help_about, onHelpAbout),
        addMenuItemListener(resourceIdentifiers::id_file_exit, onFileExit),
        addMenuItemListener(resourceIdentifiers::id_file_new, onFileNew).
% end of automatic code

end implement taskWindow

% Copyright Tsarouchas Vasileios

implement preMain inherits formWindow
```

```
    open core, vpiDomains

clauses
    display(Parent) = Form :-
        Form = new(Parent),
        Form:show().

clauses
    new(Parent) :-
        formWindow::new(Parent),
        generatedInitialize(),
        setBackgroundColor(color_lightBlue),
        setBorder(frameDecoration::thinBorder),
        _ =
            tickAction(1000,
                { :-
                    Tnow = time::now(),
                    TimeStamp = Tnow:formatDateTime("dd-MM-
yyyy", " HH:mm:ss"),
                    dateLabel:setText(TimeStamp)
                }).

predicates
    onPushButtonClick : button::clickResponder.
clauses
    onPushButtonClick(_Source) = button::defaultAction :-
        _ = frmMain::display(getParent()),
        destroy().

% This code is maintained automatically, do not update it manually.
facts
    dateLabel : textControl.
    pushButton_ctl : button.
    welcomeText : textControl.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("Welcome Form"),
        setRect(vpiDomains::rct(50, 40, 274, 160)),
        setDecoration(titlebar([frameDecoration::maximizeButton, frameDecoration
::minimizeButton, frameDecoration::closeButton])),
        setBorder(frameDecoration::sizeBorder),
        setState([wsf_ClipSiblings, wsf_ClipChildren]),
```

```
        menuSet(noMenu),
        dateLabel := textControl::new(This),
        dateLabel:setPosition(48, 56),
        dateLabel:setSize(120, 20),
        dateLabel:setAlignment(vpiDomains::alignCenter),
        dateLabel:setWrap(true),
        pushButton_ctl := button::new(This),
        pushButton_ctl:setText("Proceed"),
        pushButton_ctl:setPosition(80, 90),
        pushButton_ctl:setClickResponder(onPushButtonClick),
        welcomeText := textControl::new(This),
        welcomeText:setText("Welcome to our diagnostic tool"),
        welcomeText:setPosition(48, 12),
        welcomeText:setSize(120, 32),
        welcomeText:setBorder(true),
        welcomeText:setAlignment(vpiDomains::alignCenter).
% end of automatic code

end implement preMain

% Copyright Tsarouchas Vasileios

implement frmMain inherits formWindow
    open core, vpiDomains

clauses
    display(Parent) = Form :-
        Form = new(Parent),
        Form:show().

constants
    male = "Male".
    female = "Female".

clauses
    new(Parent) :-
        formWindow::new(Parent),
        generatedInitialize(),
        setBackgroundColor(color_lightBlue),
        setBorder(frameDecoration::thinBorder).

predicates
    onPushButtonClick : button::clickResponder.
clauses
    onPushButtonClick(_Source) = button::defaultAction :-
        Male = male_ctl:getRadioState(),
```

```
        Female = female_ctl:getRadioState(),
        if Male = radioButton::checked() then
            Patient =
                patientInfo::new(userName_ctl:getText(), surName_ctl:getText(), am
ka_ctl:getText(), phone_ctl:getText(), dateOfBirth_ctl:getText(),
                    male),
            _ = diagnose::display(This, Patient)
        elseif Female = radioButton::checked then
            Patient =
                patientInfo::new(userName_ctl:getText(), surName_ctl:getText(), am
ka_ctl:getText(), phone_ctl:getText(), dateOfBirth_ctl:getText(),
                    female),
            _ = diagnose::display(This, Patient)
        end if.

% This code is maintained automatically, do not update it manually.
facts
    pushButton_ctl : button.
    userName_ctl : editControl.
    surName_ctl : editControl.
    amka_ctl : editControl.
    dateOfBirth_ctl : editControl.
    male_ctl : radioButton.
    female_ctl : radioButton.
    phone_ctl : editControl.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("Form Main"),
        setRect(vpiDomains::rct(50, 40, 350, 206)),
        setDecoration(titlebar([frameDecoration::maximizeButton, frameDecoration
::minimizeButton, frameDecoration::closeButton])),
        setBorder(frameDecoration::sizeBorder),
        setState([wsf_ClipSiblings, wsf_ClipChildren]),
        menuSet(noMenu),
        pushButton_ctl := button::new(This),
        pushButton_ctl:setText("Submit"),
        pushButton_ctl:setPosition(108, 140),
        pushButton_ctl:setWidth(72),
        pushButton_ctl:setClickResponder(onPushButtonClick),
        GroupBox_ctl = groupBox::new(This),
        GroupBox_ctl:setText("Patient Info"),
        GroupBox_ctl:setPosition(28, 4),
```

```
        GroupBox_ctl:setWidth(244),
        GroupBox_ctl:setHeight(128),
        GroupBox_ctl:setBorderStyle(groupBox::simpleBorder),
        GroupBox_ctl:setBorder(true),
        userName_ctl := editControl::new(GroupBox_ctl),
        userName_ctl:setPosition(15, 16),
        userName_ctl:setWidth(76),
        surName_ctl := editControl::new(GroupBox_ctl),
        surName_ctl:setPosition(135, 16),
        surName_ctl:setWidth(76),
        amka_ctl := editControl::new(GroupBox_ctl),
        amka_ctl:setPosition(15, 50),
        amka_ctl:setWidth(76),
        dateOfBirth_ctl := editControl::new(GroupBox_ctl),
        dateOfBirth_ctl:setPosition(15, 86),
        dateOfBirth_ctl:setWidth(76),
        StaticText_ctl = textControl::new(GroupBox_ctl),
        StaticText_ctl:setText("Name"),
        StaticText_ctl:setPosition(15, 6),
        StaticText_ctl:setHeight(10),
        StaticText1_ctl = textControl::new(GroupBox_ctl),
        StaticText1_ctl:setText("Surname"),
        StaticText1_ctl:setPosition(135, 6),
        StaticText2_ctl = textControl::new(GroupBox_ctl),
        StaticText2_ctl:setText("SSRN"),
        StaticText2_ctl:setPosition(15, 40),
        StaticText3_ctl = textControl::new(GroupBox_ctl),
        StaticText3_ctl:setText("DOB"),
        StaticText3_ctl:setPosition(15, 76),
        male_ctl := radioButton::new(GroupBox_ctl),
        male_ctl:setText("Male"),
        male_ctl:setRect(vpiDomains::rct(135, 90, 193, 102)),
        female_ctl := radioButton::new(GroupBox_ctl),
        female_ctl:setText("Female"),
        female_ctl:setRect(vpiDomains::rct(135, 78, 193, 90)),
        phone_ctl := editControl::new(GroupBox_ctl),
        phone_ctl:setPosition(135, 50),
        phone_ctl:setWidth(72),
        StaticText4_ctl = textControl::new(GroupBox_ctl),
        StaticText4_ctl:setText("Telephone"),
        StaticText4_ctl:setPosition(135, 40).
% end of automatic code

end implement frmMain
```

```
    % Copyright Tsarouchas Vasileios

implement diagnose inherits dialog
    open core, vpiDomains

clauses
    display(Parent, Patient) = Dialog :-
        Dialog = new(Parent, Patient),
        Dialog:show().

clauses
    new(Parent, Patient) :-
        dialog::new(Parent),
        generatedInitialize(),
        setBackgroundColor(color_lightBlue),
        name_ctl:setText(Patient:getName()),
        surname_ctl:setText(Patient:getSurname()),
        amka_ctl:setText(Patient:getAMKA()),
        phone_ctl:setText(Patient:getPhone()),
        dateofbirth_ctl:setText(Patient:getDOB()),
        gender_ctl:setText(Patient:getGender()).

predicates
    onStartDiagnosisButtonClick : button::clickResponder.
clauses
    onStartDiagnosisButtonClick(_Source) = button::defaultAction :-
        diagnoseRun::run().

% This code is maintained automatically, do not update it manually.
facts
    startDiagnosisButton_ctl : button.
    name_ctl : textControl.
    surname_ctl : textControl.
    phone_ctl : textControl.
    amka_ctl : textControl.
    dateofbirth_ctl : textControl.
    gender_ctl : textControl.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("Patient Info"),
        setRect(vpiDomains::rct(50, 40, 296, 210)),
        setModal(true),
```

```
setDecoration(titlebar([frameDecoration::closeButton])),
startDiagnosisButton_ctl := button::new(This),
startDiagnosisButton_ctl:setText("Run diagnosis"),
startDiagnosisButton_ctl:setPosition(88, 152),
startDiagnosisButton_ctl:setWidth(68),
startDiagnosisButton_ctl:setClickResponder(onStartDiagnosisButtonClick),
GroupBox_ctl = groupBox::new(This),
GroupBox_ctl:setText("Patient Info"),
GroupBox_ctl:setPosition(20, 10),
GroupBox_ctl:setWidth(208),
GroupBox_ctl:setHeight(136),
GroupBox_ctl:setBorder(true),
GroupBox_ctl:setBorderStyle(groupBox::groupBox),
StaticText_ctl = textControl::new(GroupBox_ctl),
StaticText_ctl:setText("Name :"),
StaticText_ctl:setPosition(3, 8),
StaticText_ctl:setWidth(52),
StaticText1_ctl = textControl::new(GroupBox_ctl),
StaticText1_ctl:setText("Surname :"),
StaticText1_ctl:setPosition(3, 28),
StaticText1_ctl:setWidth(64),
StaticText2_ctl = textControl::new(GroupBox_ctl),
StaticText2_ctl:setText("SSRN :"),
StaticText2_ctl:setPosition(3, 70),
StaticText2_ctl:setWidth(88),
StaticText3_ctl = textControl::new(GroupBox_ctl),
StaticText3_ctl:setText("Date of Birth :"),
StaticText3_ctl:setPosition(3, 92),
StaticText3_ctl:setWidth(52),
StaticText4_ctl = textControl::new(GroupBox_ctl),
StaticText4_ctl:setText("Gender :"),
StaticText4_ctl:setPosition(3, 110),
StaticText4_ctl:setWidth(40),
StaticText5_ctl = textControl::new(GroupBox_ctl),
StaticText5_ctl:setText("Telephone :"),
StaticText5_ctl:setPosition(3, 50),
StaticText5_ctl:setWidth(60),
name_ctl := textControl::new(GroupBox_ctl),
name_ctl:setText("Static text"),
name_ctl:setPosition(107, 8),
name_ctl:setWidth(92),
surname_ctl := textControl::new(GroupBox_ctl),
surname_ctl:setText("Static text"),
surname_ctl:setPosition(107, 28),
surname_ctl:setWidth(92),
```

```
        phone_ctl := textControl::new(GroupBox_ctl),
        phone_ctl:setText("Static text"),
        phone_ctl:setPosition(107, 50),
        phone_ctl:setWidth(92),
        amka_ctl := textControl::new(GroupBox_ctl),
        amka_ctl:setText("Static text"),
        amka_ctl:setPosition(107, 70),
        amka_ctl:setWidth(92),
        dateofbirth_ctl := textControl::new(GroupBox_ctl),
        dateofbirth_ctl:setText("Static text"),
        dateofbirth_ctl:setPosition(107, 92),
        dateofbirth_ctl:setWidth(92),
        gender_ctl := textControl::new(GroupBox_ctl),
        gender_ctl:setText("Static text"),
        gender_ctl:setPosition(107, 110),
        gender_ctl:setWidth(92).
% end of automatic code

end implement diagnose

    % Copyright Tsarouchas Vasileios

implement diagnoseRun
    open core

class facts - factdb
    xpositive : (symbol, symbol).
    xnegative : (symbol, symbol) nondeterm.

clauses
    positive(X, Y) :-
        xpositive(X, Y),
        !.
    positive(X, Y) :-
        not(xnegative(X, Y)),
        question(X, Y).

clauses
    question(X, Y) :-
        !,
        Reply = answerDialog::ask(string::concat(X, " it ", Y, "\n")),
        % !,
        C = string::charLower(string::frontChar(Reply)),
        remember(X, Y, C).

clauses
```

```
    remember(X, Y, Ans) :-
        if Ans = 'y' then
            asserta(xpositive(X, Y))
        elseif Ans = 'n' then
            asserta(xnegative(X, Y)),
            fail
        end if.

    clear_facts() :-
        retractFactDb(factDb).
        %_Reply = stdio::readLine().

clauses
    run() :-
        disease(X),
        !,
        messageBox::displayNote(X, "The disease having all these symptoms is"),
        clear_facts.

    run() :-
        messageBox::displayError("The disease cannot be determined."),
        clear_facts.

clauses
    disease('Covid-19') :-
        is_disease('Covid-19').

    disease('Pharyngitis') :-
        is_disease('Pharyngitis').

    disease('Flu') :-
        is_disease('Flu').

    disease('Common Cold') :-
        is_disease('Common Cold').

    disease('Pneumonia') :-
        is_disease('Pneumonia').

    disease('Menigitis') :-
        is_disease('Menigitis').

clauses
    is_disease('Covid-19') :-
        positive('have', 'fever?'),
```

```
        positive('feel', 'chest pain?'),
        positive('have', 'lose of taste or smell?'),
        positive('have', 'cough?'),
        positive('feel', 'difficulty breathing?'),
        positive('have', 'diarrhea?'),
        positive('have', 'vomiting?').

    is_disease('Pharyngitis') :-
        positive('have', 'sudden onset of sore throat?'),
        positive('have', 'pain with swallowing?'),
        positive('have', 'fever?'),
        positive('have', 'cough?'),
        positive('have', 'rhinorrea?'),
        positive('have', 'hoarseness?'),
        positive('have', 'oral ulcers?'),
        positive('have', 'conjuctivitis?').

    is_disease('Flu') :-
        positive('have', 'fever?'),
        positive('have', 'cough?'),
        positive('have', 'sore throat?'),
        positive('have', 'runny or stuffy nose?'),
        positive('have', 'muscle or body aches?'),
        positive('have', 'headaches?'),
        positive('have', 'fatigue?').

    is_disease('Common Cold') :-
        positive('have', 'runny or stuffy nose?'),
        positive('have', 'sore throat?'),
        positive('have', 'headaches?').

    is_disease('Pneumonia') :-
        positive('have', 'fever and chills'),
        positive('have', 'cough?'),
        positive('have', 'rappid breathing or difficulty breathing?'),
        positive('have', 'chest pain?').

    is_disease('Menigitis') :-
        positive('have', 'stiff neck?'),
        positive('have', 'fever?'),
        positive('have', 'headache?'),
        positive('have', 'photophobia (eyes being more sensitive to light)?'),
        positive('have', 'confusion?').

end implement diagnoseRun
```

```
    % Copyright Tsarouchas Vasileios

implement answerDialog inherits dialog
    open core, vpiDomains

clauses
    ask(Init) = Answer :-
        Dialog = answerDialog::new(applicationWindow::get()),
        Dialog:setInitString(Init),
        Dialog:show(),
        Answer = Dialog:getAnswer().

clauses
    setInitString(InitString) :-
        question_ctl:setText(InitString),
        not(InitString = ""),
        !.

    setInitString(_).

clauses
    getAnswer() = Answer :-
        answer(Answer).

facts
    answer : (string Result) determ.

clauses
    new(Parent) :-
        dialog::new(Parent),
        generatedInitialize().

predicates
    onYesButtonClick : button::clickResponder.
clauses
    onYesButtonClick(_Source) = button::defaultAction :-
        destroy(),
        assertz(answer("yes")).

predicates
    onNoButtonClick : button::clickResponder.
clauses
    onNoButtonClick(_Source) = button::defaultAction :-
        destroy(),
        assertz(answer("no")).
```

```
% This code is maintained automatically, do not update it manually.
facts
    question_ctl : textControl.
    yesButton_ctl : button.
    noButton_ctl : button.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("answerDialog"),
        setRect(vpiDomains::rct(50, 40, 228, 120)),
        setModal(true),
        %setDecoration(titlebar([frameDecoration::closeButton])),
        setBorder(frameDecoration::noBorder),
        question_ctl := textControl::new(This),
        question_ctl:setText("question"),
        question_ctl:setPosition(28, 25),
        question_ctl:setSize(132, 8),
        question_ctl:setAlignment(vpiDomains::alignCenter),
        yesButton_ctl := button::new(This),
        yesButton_ctl:setText("Yes"),
        yesButton_ctl:setPosition(28, 42),
        yesButton_ctl:setClickResponder(onYesButtonClick),
        noButton_ctl := button::new(This),
        noButton_ctl:setText("No"),
        noButton_ctl:setPosition(104, 42),
        noButton_ctl:setClickResponder(onNoButtonClick).
% end of automatic code

end implement answerDialog

    % Copyright Tsarouchas Vasileios


    % Copyright Tsarouchas Vasileios

interface answerDialog supports dialog
    open core

predicates
    setInitString : (string InitString).

predicates
```

```
    getAnswer : () -> string Answer determ.

end interface answerDialog

implement patientInfo
    open core

facts
    name : string.
    surname : string.
    amka : string.
    phone : string.
    dateOfbirth : string.
    gender : string.

clauses
    new(Name, Surname, SSRN, Phone, DateOfBirth, Gender) :-
        name := Name,
        surname := Surname,
        amka := SSRN,
        phone := Phone,
        dateOfbirth := DateOfBirth,
        gender := Gender.

clauses
    getName() = name.

clauses
    getSurname() = surname.

clauses
    getAMKA() = amka.

clauses
    getPhone() = phone.

clauses
    getDOB() = dateOfbirth.

clauses
    getGender() = gender.

clauses
    setName(Name) :-
        name := Name.
```

```
clauses
    setSurname(Surname) :-
        surname := Surname.

clauses
    setAMKA(SSRN) :-
        amka := SSRN.

clauses
    setPhone(Phone) :-
        phone := Phone.

clauses
    setDOB(DateOfBirth) :-
        dateOfbirth := DateOfBirth.

clauses
    setGender(Gender) :-
        gender := Gender.

end implement patientInfo

    % Copyright Tsarouchas Vasileios

interface patientInfo
    open core

predicates
    % Name set,get
    getName : () -> string Name.
    setName : (string Name).
    % Surname set, get
    getSurname : () -> string Surname.
    setSurname : (string Surname).
    % AMKA set, get
    getAMKA : () -> string SSRN.
    setAMKA : (string SSRN).
    % Phone get, set
    getPhone : () -> string Phone.
    setPhone : (string Phone).
    % Date of birth set,get
    getDOB : () -> string DateOfBirth.
    setDOB : (string DateOfBirth).
    % Gender set,get
    getGender : () -> string Gender.
```

```
    setGender : (string Gender).

end interface patientInfo
```

Java

```
package thesisuowm;


import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import thesisuowm.Forms.FrmMain;


/**

 *

 * @author chala

 */

public class ThesisUOWM extends Application

{

   private static final int COUNT_LIMIT = 10;


   @Override

   public void start(Stage primaryStage) throws Exception

   {

      Parent root = FXMLLoader.load(getClass().getResource("Welcome.fxml"));

      Scene scene = new Scene(root);

      primaryStage.setScene(scene);

      primaryStage.show();

      root.getScene().getWindow().hide();
```

```
}

@Override
public void init() throws Exception
{
   try
   {
   for (int i = 0; i < COUNT_LIMIT; i++)
   {
      double progress = (double)i/10;
      notifyPreloader(new Loader.ProgressNotification(progress));
      Thread.sleep(500);
   }
      //Login Form
      FrmMain main = new FrmMain();
      main.start();
   }
   catch(Exception ex)
   {
      System.out.println(ex.getMessage());
   }
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args)
```

```
    {
        System.setProperty("javafx.preloader", Loader.class.getCanonicalName());
        Application.launch(ThesisUOWM.class, args);
    }
}
package thesisuowm;

import javafx.application.Preloader;
import static javafx.application.Preloader.StateChangeNotification.Type.BEFORE_START;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

/**
 *
 * @author chala
 */
public class Loader extends Preloader
{
    private Stage preloaderStage;
    private Scene scene;

    public Loader()
    {
```

```
    }

    @Override
    public void init() throws Exception
    {
       Parent root = FXMLLoader.load(getClass().getResource("Welcome.fxml"));
       scene = new Scene(root);

    }

    @Override
    public void start(Stage primaryStage) throws Exception
    {
       this.preloaderStage = primaryStage;
       this.preloaderStage.setScene(scene);
       this.preloaderStage.initStyle(StageStyle.UNDECORATED);
       this.preloaderStage.show();

    }

     @Override
     public void handleApplicationNotification(Preloader.PreloaderNotification info)
     {
        if(info instanceof Preloader.ProgressNotification)
        {
           WelcomeController.progLabel.setText(((Preloader.ProgressNotification)
info).getProgress()*100 + "%");
```

```
            WelcomeController.startprogress.setProgress(((Preloader.ProgressNotification)
info).getProgress());
          }
        }

        @Override
        public void handleStateChangeNotification(Preloader.StateChangeNotification info)
        {
           Preloader.StateChangeNotification.Type type = info.getType();
           switch(type)
           {
              case BEFORE_START:
                 preloaderStage.hide();
                 break;
           }
        }
    }

    package thesisuowm;

    import java.net.URL;
    import java.util.ResourceBundle;
    import javafx.fxml.FXML;
    import javafx.fxml.Initializable;
    import javafx.scene.control.Label;
    import javafx.scene.control.ProgressBar;

    /**
```

```
 * FXML Controller class
 *
 * @author chala
 */
public class WelcomeController implements Initializable {



    @FXML
    private ProgressBar progress;
    public static ProgressBar startprogress;
    @FXML
    private Label label;
    public static Label progLabel;


    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb)
    {

progress.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/progressBarStyle.css").toExte
rnalForm());


        startprogress = progress;
        progLabel = label;
    }
```

```
}
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ProgressBar?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" prefHeight="349.0" prefWidth="977.0"
xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="thesisuowm.WelcomeController">
  <children>
    <ImageView fitHeight="400.0" fitWidth="977.0" pickOnBounds="true" preserveRatio="true">
      <image>
        <Image url="@Icons/kastoria-uowm-1024x373.png" />
      </image>
    </ImageView>
    <ProgressBar fx:id="progress" layoutX="19.0" layoutY="318.0" prefHeight="24.0"
prefWidth="940.0" progress="0.0" style="-fx-border-color: transparent;"/>
    <Label fx:id="label" alignment="CENTER" contentDisplay="CENTER" layoutX="448.0"
layoutY="321.0" prefHeight="18.0" prefWidth="55.0" text="Label" textAlignment="CENTER">
      <font>
        <Font name="Arial" size="10.0" />
      </font></Label>
  </children>
</AnchorPane>
package thesisuowm.Classes;
```

```
/**
 *
 * @author chala
 */
public class Patient
{
  private String name,surname,gender,dateOfBirth,AMKA,number;
  private int userID;

  public Patient(int userID ,String name, String surname, String AMKA, String gender, String dateOfBirth, String number )
   {
    this.name = name;
    this.surname = surname;
    this.AMKA = AMKA;
    this.dateOfBirth = dateOfBirth;
    this.gender = gender;
    this.number = number;
    this.userID = userID;
   } //constructor

  public String getName(){return name;}

  public String getSurname() {return surname;}

  public String getAMKA() {return AMKA;}
```

```
  public String getGender() {return gender;}


  public String getDateOfBirth() {return dateOfBirth;}


  public String getNumber() {return number;}


  public int getUserID() {return userID;}


  @Override
  public String toString()
  {
   return "Name : " + name + "\n" +
         "Surname : " + surname + "\n" +
         "Social Security Registration Number : " + AMKA + "\n" +
         "Gender : " + gender + "\n" +
         "Date of Birth : " + dateOfBirth + "\n" +
         "Phone Number : " + number + "\n";
  }


}
package thesisuowm.Classes;


import java.util.Date;


/**
 *
 * @author chala
```

```
 */
public class PatientRecord
{
   private int userID;
   private String name,surname,condition,diagnosisDate;

   public PatientRecord(int patientID, String name, String surname, String condition, String
diagnosisDate)
   {
      this.userID = patientID;
      this.name = name;
      this.surname = surname;
      this.condition = condition;
      this.diagnosisDate = diagnosisDate;
   }

   // Getters and setters for the properties
   public int getUserID()
   {
      return userID;
   }

   public void setPatientID(int patientID)
   {
      this.userID = patientID;
   }

   public String getName()
```

```
{
   return name;
}

public void setName(String name)
{
   this.name = name;
}

public String getSurname()
{
   return surname;
}

public void setSurname(String surname)
{
   this.surname = surname;
}

public String getCondition()
{
   return condition;
}

public void setCondition(String condition)
{
   this.condition = condition;
```

```
    }

    public String getDiagnosisDate()
    {
        return diagnosisDate;
    }

    public void setDiagnosisDate(String diagnosisDate)
    {
        this.diagnosisDate = diagnosisDate;
    }
}
package thesisuowm.Classes;

/**
 *
 * @author chala
 */
public class User
{
    private String name,surname,username,password,email;
    private int ID = 0;

    public User(String name, String surname, String username, String password, String email)
    {
        this.name = name;
        this.surname = surname;
```

```
        this.username = username;
        this.password = password;
        this.email = email;
    }//Constructor

    public User() {
        throw new UnsupportedOperationException("Not supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
    }

    //Set
    public void setID(int id)
    {
        this.ID = id;
    }
    //Get
    public String getName() {return name;}
    public String getSurname() {return surname;}
    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public String getEmail() {return email;}
    public int getID() {return ID;}
}
package thesisuowm.Controls;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
```

```
import java.util.stream.Collectors;
import javafx.collections.FXCollections;
import javafx.collections.ListChangeListener;
import javafx.collections.ObservableList;
import javafx.collections.ObservableSet;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.geometry.Side;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.CustomMenuItem;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Priority;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.scene.text.TextFlow;

/**
 *
 * @author chala
 */
public class AutoCompleteMultiSelectionBox extends HBox
```

```
{
    private final ObservableList<String> tags;
    private final ObservableSet<String> suggestions;
    private ContextMenu entriesPopup;
    private static final int MAX_ENTRIES = 10 ;

    private final TextField inputTextField;

    public AutoCompleteMultiSelectionBox() {
        getStyleClass().setAll("tag-bar");
        getStylesheets().add(getClass().getResource("/thesisuowm/Styles/style.css").toExternalForm());
        tags = FXCollections.observableArrayList();
        suggestions = FXCollections.observableSet();
        inputTextField = new TextField();
        this.entriesPopup = new ContextMenu();
        setListner();
        inputTextField.setOnKeyPressed(event -> {
            // Remove last element with backspace
            if (event.getCode().equals(KeyCode.BACK_SPACE) && !tags.isEmpty() &&
inputTextField.getText().isEmpty()) {
                String last = tags.get(tags.size() - 1);
                suggestions.add(last);
                tags.remove(last);
            }
        });

        inputTextField.prefHeightProperty().bind(this.heightProperty());
        inputTextField.setPromptText("Type your symptoms here!");
```

```
        HBox.setHgrow(inputTextField, Priority.ALWAYS);
        inputTextField.setBackground(null);

        tags.addListener((ListChangeListener.Change<? extends String> change) -> {
           while (change.next()) {
              if (change.wasPermutated()) {
                 ArrayList<Node> newSublist = new ArrayList<>(change.getTo() - change.getFrom());
                 for (int i = change.getFrom(), end = change.getTo(); i < end; i++) {
                    newSublist.add(null);
                 }
                 for (int i = change.getFrom(), end = change.getTo(); i < end; i++) {
                    newSublist.set(change.getPermutation(i), getChildren().get(i));
                 }
                 getChildren().subList(change.getFrom(), change.getTo()).clear();
                 getChildren().addAll(change.getFrom(), newSublist);
              } else {
                 if (change.wasRemoved()) {
                    getChildren().subList(change.getFrom(), change.getFrom() +
change.getRemovedSize()).clear();
                 }
                 if (change.wasAdded()) {
                    getChildren().addAll(change.getFrom(),
change.getAddedSubList().stream().map(Tag::new).collect(
                          Collectors.toList()));
                 }
              }
           }
        });
```

```
    getChildren().add(inputTextField);
  }

  /**
   * Build TextFlow with selected text. Return "case" dependent.
   *
   * @param text   - string with text
   * @param filter - string to select in text
   * @return - TextFlow
   */
  private static TextFlow buildTextFlow(String text, String filter) {
    int filterIndex = text.toLowerCase().indexOf(filter.toLowerCase());
    Text textBefore = new Text(text.substring(0, filterIndex));
    Text textAfter = new Text(text.substring(filterIndex + filter.length()));
    Text textFilter = new Text(text.substring(filterIndex,
        filterIndex + filter.length())); //instead of "filter" to keep all "case sensitive"
    textFilter.setFill(Color.ORANGE);
    textFilter.setFont(Font.font("Helvetica", FontWeight.BOLD, 12));
    return new TextFlow(textBefore, textFilter, textAfter);
  }

  /**
   * "Suggestion" specific listners
   */
  private void setListner() {
    //Add "suggestions" by changing text
    inputTextField.textProperty().addListener((observable, oldValue, newValue) -> {
```

```
        //always hide suggestion if nothing has been entered (only "spacebars" are dissalowed in
TextFieldWithLengthLimit)
        if (newValue.isEmpty()) {
            entriesPopup.hide();
        } else {
            //filter all possible suggestions depends on "Text", case insensitive
            List<String> filteredEntries = suggestions.stream()
                    .filter(e -> e.toLowerCase().contains(newValue.toLowerCase()))
                    .collect(Collectors.toList());
            //some suggestions are found
            if (!filteredEntries.isEmpty()) {
                //build popup - list of "CustomMenuItem"
                populatePopup(filteredEntries, newValue);
                if (!entriesPopup.isShowing()) { //optional
                    entriesPopup.show(this, Side.BOTTOM, 0, 0); //position of popup
                }
                //no suggestions -> hide
            } else {
                entriesPopup.hide();
            }
        }
    });

    //Hide always by focus-in (optional) and out
    focusedProperty().addListener((observableValue, oldValue, newValue) -> entriesPopup.hide());
}

/**
```

```
     * Populate the entry set with the given search results. Display is limited to 10 entries, for
performance.
     *
     * @param searchResult The set of matching strings.
     */
    private void populatePopup(List<String> searchResult, String searchRequest) {
      //List of "suggestions"
      List<CustomMenuItem> menuItems = new LinkedList<>();
      //Build list as set of labels
      searchResult.stream()
          .limit(MAX_ENTRIES) // Limit to MAX_ENTRIES in the suggestions
          .forEach((String result) -> {
            //label with graphic (text flow) to highlight founded subtext in suggestions
            TextFlow textFlow = buildTextFlow(result, searchRequest);

textFlow.prefWidthProperty().bind(AutoCompleteMultiSelectionBox.this.widthProperty());
            CustomMenuItem item = new CustomMenuItem(textFlow, true);
            menuItems.add(item);

            //if any suggestion is select set it into text and close popup
            item.setOnAction(actionEvent -> {
              tags.add(result);
              suggestions.remove(result);
              inputTextField.clear();
              entriesPopup.hide();
            });
          });
```

```
    //"Refresh" context menu
    entriesPopup.getItems().clear();
    entriesPopup.getItems().addAll(menuItems);
}

public final ObservableList<String> getTags() {
    return tags;
}

public final ObservableSet<String> getSuggestions() {
    return suggestions;
}

/**
 * Clears then repopulates the entries with the new set of data.
 *
 * @param suggestions set of items.
 */

public final void setSuggestions(ObservableSet<String> suggestions) {
    this.suggestions.clear();
    this.suggestions.addAll(suggestions);
}

private class Tag extends HBox {
    Tag(String tag) {
        // Style
```

```
            getStyleClass().add("tag");

            // Remove item button
            Button removeButton = new Button("X");
            removeButton.setBackground(null);
            removeButton.setOnAction(event -> {
               tags.remove(tag);
               suggestions.add(tag);
               inputTextField.requestFocus();
            });

            // Displayed text
            Text text = new Text(tag);
            text.setFill(Color.BLACK);
            text.setFont(Font.font(text.getFont().getFamily(), FontWeight.LIGHT, text.getFont().getSize()));

            // Children position
            setAlignment(Pos.CENTER);
            setSpacing(1.5);
            setPadding(new Insets(0, 0, 0, 2));

            getChildren().addAll(text, removeButton);
         }
      }
   }
   /*
    * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package thesisuowm.Database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.scene.control.Alert;
import java.sql.*;
import thesisuowm.Classes.Patient;
import thesisuowm.Classes.PatientRecord;
import thesisuowm.Classes.User;



/**
 *
 * @author chala
 */
public class DatabaseService
{
   private static Connection conn = null;

   public static void Connect() throws ClassNotFoundException
   {
      try
      {
```

```
            Class.forName("org.sqlite.JDBC");
            conn = DriverManager.getConnection("jdbc:sqlite:src/thesisuowm/Database/db.sqlite");
            System.out.println("Connected to SQLite database successfully.");
        } catch (SQLException e)
        {
            System.out.println(e.getMessage());
        }
    }

    public static void CreateUser(User user)
    {
        String name = user.getName();
        String surname = user.getSurname();
        String username = user.getUsername();
        String email = user.getEmail();
        String password = user.getPassword();
        String confirmPassword = user.getPassword();

        if (username.isEmpty() || email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()
|| name.isEmpty() || surname.isEmpty())
        {
            // Display an error message if any fields are empty
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Please fill in all fields");
            alert.showAndWait();
            return;
        }
```

```
        if (!password.equals(confirmPassword))
        {
            // Display an error message if the password and confirm password fields do not match
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Passwords do not match");
            alert.showAndWait();
            return;
        }

        try
        {

            // Create a prepared statement to insert the new user record
            PreparedStatement stmt = conn.prepareStatement("INSERT INTO users (name, surname,
username, password, email) VALUES (?, ?, ?, ?, ?)");
            stmt.setString(1, name);
            stmt.setString(2, surname);
            stmt.setString(3, username);
            stmt.setString(4, password);
            stmt.setString(5, email);

            // Execute the prepared statement to insert the new user record
            stmt.executeUpdate();

            // Display a success message
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
```

```
            alert.setTitle("Success");
            alert.setHeaderText("User created successfully");
            alert.showAndWait();
        }
        catch (SQLException e)
        {
            // Display an error message if there was a problem inserting the user record
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("An error occurred while creating the user");
            alert.setContentText(e.getMessage());
            alert.showAndWait();
        }
    }

    public static User Login(String username, String password)
    {
    try {
        PreparedStatement stmt = conn.prepareStatement("SELECT * FROM users WHERE username = ?
and password = ?");
        stmt.setString(1, username);
        stmt.setString(2, password);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            String _passwd = rs.getString("password");
            String _username = rs.getString("username");
            if (password.equals(_passwd) && username.equals(_username))
```

```
        {
            String id = rs.getString("id");
            String name = rs.getString("name");
            String surname = rs.getString("surname");
            String email = rs.getString("email");
            User user = new User(name, surname, username, password, email);
            user.setID(Integer.parseInt(id));
            System.out.println("Login successful");
            rs.close();
            stmt.close();
            return user;
        } else
        {
            System.out.println("Invalid password");
        }
    } else {
        System.out.println("User not found. Try again or create an account");
    }

    rs.close();
    stmt.close();
}
catch (SQLException e)
{
    System.err.println("SQL error: " + e.getMessage());
}
```

```
        return null;
    }

    public static void Insert_Or_Update(Patient patient, boolean patientExists)
    {
        if(patientExists)
        {
            Update(patient);
        }
        else
        {
            Insert(patient);
        }
    }

    private static void Insert(Patient patient)
    {
        try
        {
            String query = "INSERT INTO patients (user_id, name, surname, amka, telephone, gender, dob)
VALUES (?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setInt(1, patient.getUserID());
            stmt.setString(2, patient.getName());
            stmt.setString(3, patient.getSurname());
            stmt.setString(4, patient.getAMKA());
            stmt.setString(5, patient.getNumber());
            stmt.setString(6, patient.getGender());
```

```
            stmt.setString(7, patient.getDateOfBirth());
            stmt.executeUpdate();
        }
        catch (SQLException ex)
        {
            System.out.println("Error creating patient: " + ex.getMessage());
        }
    }

    private static void Update(Patient patient)
    {
        try
        {
            String query = "UPDATE patients SET name = ?, surname = ?, amka = ?, telephone = ?, gender =
?, dob = ? WHERE user_id = ?";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setString(1, patient.getName());
            stmt.setString(2, patient.getSurname());
            stmt.setString(3, patient.getAMKA());
            stmt.setString(4, patient.getNumber());
            stmt.setString(5, patient.getGender());
            stmt.setString(6, patient.getDateOfBirth());
            stmt.setInt(7, patient.getUserID());

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected == 0)
            {
                throw new SQLException("Updating patient failed, no rows affected.");
```

```
        }
    }
    catch (SQLException ex)
    {
        System.out.println("Error updating patient: " + ex.getMessage());
    }
}

public static boolean checkIfPatientExists(int userId)
{
    try
    {
        String query = "SELECT COUNT(*) FROM patients WHERE user_id = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setInt(1, userId);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            int count = rs.getInt(1);
            return count > 0;
        }
        else
        {
            return false;
        }
    } catch (SQLException ex)
    {
        System.out.println("Error checking if patient exists: " + ex.getMessage());
```

```
        return false;
    }
}

public static Patient getPatientByUserId(int userId) {
    try
    {
        String query = "SELECT * FROM patients WHERE user_id = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setInt(1, userId);
        ResultSet rs = stmt.executeQuery();
        if (rs.next())
        {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            String surname = rs.getString("surname");
            String ssn = rs.getString("amka");
            String phone = rs.getString("telephone");
            String gender = rs.getString("gender");
            String dateOfBirth = rs.getString("dob");
            Patient patient = new Patient(id, name, surname, ssn, gender, dateOfBirth, phone);
            return patient;
        }
        else
        {
            return null;
        }
```

```
        }
        catch (SQLException ex)
        {
            System.out.println("Error retrieving patient: " + ex.getMessage());
            return null;
        }
    }

    public static boolean saveRecord(PatientRecord record)
    {

        try
        {
            String query = "INSERT INTO patient_records (patientID, name, surname, condition,
diagnosisDate) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(query);
            stmt.setInt(1, record.getUserID());
            stmt.setString(2, record.getName());
            stmt.setString(3, record.getSurname());
            stmt.setString(4, record.getCondition());
            stmt.setString(5, record.getDiagnosisDate());
            stmt.executeUpdate();
            return true;
        }
        catch (SQLException ex)
        {
            System.out.println("Error creating patient: " + ex.getMessage());
            return false;
```

```
        }
    }
}
package thesisuowm.Forms;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.scene.text.TextAlignment;
import javafx.stage.Stage;

/**
 *
 * @author chala
 */
public class FrmAbout
{
    private Pane window;
    private Text appInfo,studentInfo;
```

```
private Label professorInfo;
private TextArea text;
ImageView image;

public FrmAbout()
{
   this.window = new Pane();
   this.appInfo = new Text("App Info");
   this.studentInfo = new Text("Student Info");
   this.professorInfo = new Label("Professor Info");
   this.image = new ImageView();
   this.text = new TextArea();
   SetProperties();
}

public void Show()
{

   Scene scene = new Scene(window);

   Stage primaryStage = new Stage();
   primaryStage.setResizable(false);
   primaryStage.setHeight(550);
   primaryStage.setWidth(900);
   primaryStage.setScene(scene);
   primaryStage.setTitle("Medical Diagnostic Tool");
   Image icon = new Image(getClass().getResourceAsStream("/thesisuowm/Icons/more-info.png"));
```

```
    primaryStage.getIcons().add(icon);
    primaryStage.show();
}

private void SetProperties()
{
    this.professorInfo.setText("Επιβλεπων καθηγητης : Κ. Παναγιώτης Μπάτος");
    this.professorInfo.setLayoutX(14);
    this.professorInfo.setLayoutY(489);

    SetWindowProps();
    SetTextProperties();
    SetTextAreaProps();
    SetImageViewProps();
}

private void SetWindowProps()
{
     // window properties
    window.setStyle("-fx-background-color: white");
    window.getChildren().addAll(appInfo,studentInfo,professorInfo,image,text);



}

private void SetTextProperties()
{
```

```
        this.appInfo.setText("Σχεδιασμός-Δημιουργία ιατρικής διαγνωστικής εφαρμογής με την χρήση της γλώσσας Java");
        this.appInfo.setLayoutX(177);
        this.appInfo.setLayoutY(139);
        this.appInfo.textAlignmentProperty().set(TextAlignment.CENTER);
        this.appInfo.setFont(new Font("Segoe UI Semilight Italic",25));
        this.appInfo.wrappingWidthProperty().set(536.541015625);

        this.studentInfo.setText("Μια εργασία απο τους φοιτητές Τσαρουχά Βασίλειο και Λουκούμη Ιωάννα για το Πανεπιστήμιο Δυτικής Μακεδονίας.");
        this.studentInfo.setLayoutX(170);
        this.studentInfo.setLayoutY(236);
        this.studentInfo.textAlignmentProperty().set(TextAlignment.CENTER);
        this.studentInfo.setFont(new Font("Segoe UI Semilight Italic",14));
        this.studentInfo.wrappingWidthProperty().set(550.541015625);
    }

    private void SetTextAreaProps()
    {
        this.text.setLayoutX(173);
        this.text.setLayoutY(275);
        this.text.setPrefSize(537, 173);
        this.text.editableProperty().set(false);
        this.text.wrapTextProperty().set(true);

        File file = new File(getClass().getResource("/thesisuowm/aboutApp.txt").getFile());

        try (Scanner input = new Scanner(file))
```

```
        {
            while (input.hasNext())
            {
                text.appendText(input.nextLine());
            }
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        }
        text.requestFocus();
    }

    private void SetImageViewProps()
    {
        //image view properties
        image.setImage(new Image(getClass().getResourceAsStream("/thesisuowm/Icons/univ-west-maced.png")));
        image.setFitHeight(110);
        image.setFitWidth(439);
        image.setLayoutX(226);
        image.setLayoutY(14);
        image.pickOnBoundsProperty().set(true);
        image.preserveRatioProperty().set(true);
    }
}
package thesisuowm.Forms;

import thesisuowm.KnowledgeBase.RuleEngine;
import java.io.PrintStream;
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.ObservableSet;
import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Dialog;
import javafx.scene.control.ListView;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import thesisuowm.Classes.Patient;
import thesisuowm.Classes.PatientRecord;
import thesisuowm.Controls.AutoCompleteMultiSelectionBox;
import thesisuowm.Database.DatabaseService;

/**
```

```
 *
 * @author chala
 */
public class FrmDiagnosis
{
  private Stage primaryStage;
  private Patient patient;
  private ScrollPane scroll;
  private AnchorPane panel;
  private ObservableSet<String> suggestions;
  private AutoCompleteMultiSelectionBox box;
  private TextArea console;
  private Button btn_submit;
  private PrintStream ps;
  private ListView<Patient> listView;
  private ObservableList<Patient> patientObservableList;

   public FrmDiagnosis()
   {
      this.scroll = new ScrollPane();
      this.panel = new AnchorPane();
      this.suggestions = FXCollections.observableSet();
      this.box = new AutoCompleteMultiSelectionBox();
      this.console = new TextArea();
      this.listView = new ListView();
      this.patientObservableList = FXCollections.observableArrayList();
      this.btn_submit = new Button();
```

```
        SetProperties();


    }


    public void Show(Patient patient) throws Exception
    {
        primaryStage = new Stage();
        Scene scene = new Scene(panel);
        primaryStage.setScene(scene);
        primaryStage.setResizable(false);
        primaryStage.setTitle("Diagnosis Form");
        Image icon = new Image(getClass().getResourceAsStream("/thesisuowm/Icons/symptoms-
check.png"));
        primaryStage.getIcons().add(icon);
        patientObservableList.add(patient);
        listView.setItems(patientObservableList);
        this.patient = patient;
        primaryStage.show();


        // create a new thread to run the setWelcomeMessage() method
        Thread thread = new Thread(() ->
        {
            setWelcomeMessage();
        });


        // start the thread
        thread.start();
```

```
}

private void SetProperties()
{
    //AnchorPane properties
    panel.setPrefSize(660, 452);
    panel.prefHeight(452.0);
    panel.prefWidth(640.0);
    panel.setStyle("-fx-background-color: #6694de");
    panel.getChildren().add(console);
    panel.getChildren().add(btn_submit);
    //panel.getChildren().add(box);
    panel.getChildren().add(listView);
    panel.getChildren().add(scroll);

    //Textfield properties
    console.setPrefSize(528, 150);
    console.setLayoutX(56.0);
    console.setLayoutY(150.0);
    console.editableProperty().set(false);
    console.setStyle("-fx-highlight-fill: lightgray; -fx-highlight-text-fill: firebrick; -fx-font-size: 12px;");
    console.setScrollTop(Double.MAX_VALUE);

    scroll.setPrefSize(528,50);
    scroll.setLayoutX(58);
    scroll.setLayoutY(340);
    scroll.setContent(box);
```

```
        scroll.setVbarPolicy(ScrollPane.ScrollBarPolicy.NEVER);

scroll.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/scrollStyle.css").toExternalForm()
);


        listView.setPrefSize(528, 110);
        listView.setLayoutX(56);
        listView.setLayoutY(32);
        listView.prefHeight(98);
        listView.prefWidth(528);

listView.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/listStyle.css").toExternalForm()
);



        fillSuggestions();
        //AutoComplete textfield
        box.setPrefSize(520,50);
        box.setSuggestions(suggestions);
        box.autosize();
        box.setFillHeight(false);



        //button properties
        btn_submit.setAlignment(Pos.CENTER);
        btn_submit.setDefaultButton(true);
        btn_submit.setPrefSize(106.0, 30.0);
        btn_submit.setLayoutX(267);
        btn_submit.setLayoutY(390);
```

```
        btn_submit.setText("Submit");
        btn_submit.mnemonicParsingProperty().set(false);
        btn_submit.setStyle("-fx-background-color: #fc5a03");

btn_submit.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/ClickEffectStyle.css").toExt
ernalForm());
        btn_submit.setTextFill(Color.BLACK);
        btn_submit.setCursor(Cursor.HAND);
        btn_submit.setOnMouseClicked(event ->
        {
           try {
              if(box.getTags().isEmpty())
              {
                 throw new IllegalArgumentException("No symptoms provided.");
              }

              List<String> symptoms = box.getTags();
              GetDiagnosis(symptoms);
           }
           catch(Exception ex)
           {
              System.err.println(ex.getMessage());
           }
        });
     }

     private void GetDiagnosis(List<String> symptoms)
     {
```

```
    if(symptoms.size() > 0)
    {
       RuleEngine engine = new RuleEngine();
       String diagnosis = engine.execute(symptoms);

       if (diagnosis.equals("unrecognized"))
       {
          // No diagnosis found
          Alert alert = new Alert(Alert.AlertType.WARNING);
          alert.setTitle("No Diagnosis Found");
          alert.setHeaderText("No diagnosis was found for the specified symptoms");
          alert.setContentText("Please consult a doctor.");
          alert.showAndWait();
       }
       else
       {
          informUser(diagnosis);
          saveRecord(diagnosis);
       }

    }
}

 private void informUser(String diagnosis)
 {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Symptom Checker");
```

```
    alert.setHeaderText("Diagnosis result");
    alert.setContentText(diagnosis);
    alert.showAndWait();
}

private void saveRecord(String diagnosis)
{
    Dialog<ButtonType> dialog = new Dialog<>();
    dialog.setContentText("Save record to database?");
    dialog.getDialogPane().getButtonTypes().addAll(ButtonType.YES, ButtonType.NO);
    dialog.showAndWait().ifPresent(response ->
    {
        if (response == ButtonType.YES)
        {
            PatientRecord record = new PatientRecord
            (
                this.patient.getUserID(),
                this.patient.getName(),
                this.patient.getSurname(),
                diagnosis,
                LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss"))

            );

            if(DatabaseService.saveRecord(record))
            {
                Thread thread = new Thread(() ->
```

```
            {
                confirmationMessage(diagnosis);
            });

            thread.run();
          }

      } else
      {
          primaryStage.close();
      }
   });
}

private void fillSuggestions()
{
   suggestions.add("Fever or chills");
   suggestions.add("Cough");
   suggestions.add("Fatigue");
   suggestions.add("Headache");
   suggestions.add("Loss of taste or smell");
   suggestions.add("Aches");
   suggestions.add("Sneezing");
   suggestions.add("Runny or Stuffy nose");
   suggestions.add("Sore Throat");
   suggestions.add("Ear pain");
   suggestions.add("Difficulty breathing");
```

```
        suggestions.add("Confusion");
        suggestions.add("Stiff Neck");
        suggestions.add("Chest Pain");
        suggestions.add("Odynophagia");
        suggestions.add("Diarrhea");
        suggestions.add("Nausea or Vomiting");
        suggestions.add("Rhinorrhea");
        suggestions.add("Hoarseness");
        suggestions.add("Conjunctivitis");
        suggestions.add("Oral Ulcers");
        suggestions.add("Photophobia");
        suggestions.add("Confusion");
    }
    private void setWelcomeMessage()
    {
        String welcomeMessage = "Dear " + patient.getName() + "\n"
            + "Welcome to the diagnosis form! \n"
            + "Please enter your symptoms below so that we can better understand your condition\nand
provide you with the best possible treatment. \n"
            + "Don't worry,\nYour privacy is our top priority and all information you provide will be kept
confidential. \n"
            + "Thank you for entrusting us with your care.\n";

        // simulate typing effect
        btn_submit.setDisable(true);
        scroll.setDisable(true);
        for (int i = 0; i < welcomeMessage.length(); i++)
        {
```

```
            try
            {
               Thread.sleep(50); // delay between each character append
            } catch (InterruptedException e)
            {
               e.printStackTrace();
            }

            console.appendText(String.valueOf(welcomeMessage.charAt(i)));
         }
         btn_submit.setDisable(false);
         scroll.setDisable(false);
      }

      private void confirmationMessage(String diagnosis)
      {
         console.clear();
         String confirmMessage = String.format("Thank you for submitting your diagnosis: %n%s",
diagnosis);

         for (int i = 0; i < confirmMessage.length(); i++)
         {
            try
            {
               Thread.sleep(150); // delay between each character append
            } catch (InterruptedException e)
            {
               e.printStackTrace();
```

```
            }


            console.appendText(String.valueOf(confirmMessage.charAt(i)));
        }
    }
}
package thesisuowm.Forms;


import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import javafx.animation.Animation;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.application.Platform;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyCode;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
```

```
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import javafx.util.Duration;
import thesisuowm.Database.DatabaseService;

/**
 *
 * @author chala
 */
public class FrmMain extends Thread
{
   Button btn_proceed = new Button("Login");
   Label lbl_about = new Label("About");
   Label lbl_register = new Label("Sing Up");
   Label lbl_uowm = new Label("UNIVERSITY OF WESTERN MACEDONIA");
   Label time = new Label();
   ImageView image = new ImageView();
   TextField txtUsername = new TextField();
   PasswordField passwdBox = new PasswordField();
   Pane FrmMain = new Pane();


   @Override
   public void run()
   {
      try
```

```
        {
            Platform.runLater(new Runnable()
            {
                @Override
                public void run()
                {
                    try
                    {
                        SetProperties();
                        Scene scene = new Scene(FrmMain);
                        Stage primaryStage = new Stage();

                        primaryStage.setResizable(false);
                        primaryStage.setHeight(400);
                        primaryStage.setWidth(800);
                        primaryStage.setScene(scene);
                        primaryStage.setTitle("Medical Diagnostic Tool");
                        Image icon = new
Image(getClass().getResourceAsStream("/thesisuowm/Icons/medical.png"));
                        primaryStage.getIcons().add(icon);
                        primaryStage.show();
                        DatabaseService.Connect();
                    }
                    catch(Exception ex) {System.out.println(ex.getMessage());}
                }
            });
        }
        catch(Exception ex){}
```

```
    }
     private void SetProperties()
    {
       //Form properties
       FrmMain.getChildren().add(lbl_about);
       FrmMain.getChildren().add(btn_proceed);
       FrmMain.getChildren().add(image);
       FrmMain.getChildren().add(lbl_uowm);
       FrmMain.getChildren().add(time);
       FrmMain.getChildren().add(txtUsername);
       FrmMain.getChildren().add(passwdBox);
       FrmMain.getChildren().add(lbl_register);

       FrmMain.setStyle("-fx-background-color: #6694de");

       ButtonProperties();
       ImageViewProperties();
       LabelProperties();
       TextfieldsProperties();
       LabelRegisterProperties();
    }

    private void ImageViewProperties()
    {
       //image view properties
       image.setImage(new Image(getClass().getResourceAsStream("/thesisuowm/Icons/uowm-logo-
big-removebg-preview.png")));
       image.setFitHeight(62);
```

```
    image.setFitWidth(64);
    image.setLayoutX(152);
    image.setLayoutY(37);
    image.pickOnBoundsProperty().set(true);
    image.preserveRatioProperty().set(true);
}

private void LabelProperties()
{
   //Label Properties
  lbl_about.setLayoutX(730.0);
  lbl_about.setLayoutY(330.0);
  lbl_about.setPrefSize(33.0, 22.0);
  lbl_about.setTextFill(Color.BLACK);

   // event handler to change color when mouse enters
  EventHandler<MouseEvent> labelAboutEnterHandler = (MouseEvent event) ->
  {
     lbl_about.setTextFill(Color.RED);
     lbl_about.setCursor(Cursor.HAND);
  };
  lbl_about.setOnMouseEntered(labelAboutEnterHandler);

  // event handler to change color when mouse exits
  EventHandler<MouseEvent> labelAboutExitHandler = new EventHandler<MouseEvent>()
  {
     @Override
```

```
            public void handle(MouseEvent event) {
                lbl_about.setTextFill(Color.BLACK);
            }
        };
        lbl_about.setOnMouseExited(labelAboutExitHandler);
        lbl_about.setOnMouseClicked(onClick ->
        {
            FrmAbout about = new FrmAbout();
            about.Show();
        });

        // label uowm properties
        lbl_uowm.setPrefSize(447.0, 62.0);
        lbl_uowm.setLayoutX(209);
        lbl_uowm.setLayoutY(37);
        lbl_uowm.setFont(new Font("Segoe UI Light Italic",25));

         //set label time
         time.setPrefSize(350, 75);
         time.setLayoutX(280);
         time.setLayoutY(80);
         time.setFont(new Font("Segoe UI Light Italic",26));
         time.setTextFill(Color.valueOf("#ffae00"));
         Timeline clock = new Timeline(new KeyFrame(Duration.ZERO, e ->
         time.setText(LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd-MMM-yyyy
HH:mm:ss")))),
         new KeyFrame(Duration.seconds(1)));
         clock.setCycleCount(Animation.INDEFINITE);
```

```
    clock.play();
}

private void TextfieldsProperties()
{
  txtUsername.setPrefWidth(200);
  txtUsername.setLayoutX(300);
  txtUsername.setLayoutY(180);
  txtUsername.setPromptText("Username");
  txtUsername.setAlignment(Pos.CENTER);

  passwdBox.setPrefWidth(200);
  passwdBox.setLayoutX(300);
  passwdBox.setLayoutY(220);
  passwdBox.setPromptText("Password");
  passwdBox.setAlignment(Pos.CENTER);

  txtUsername.setOnKeyPressed(e ->
  {
     if(e.getCode() == KeyCode.ENTER)
     {
        ExecuteLogin();
     }
  });

  passwdBox.setOnKeyPressed(e ->
  {
```

```
        if(e.getCode() == KeyCode.ENTER)
        {
            ExecuteLogin();
        }
    });
}

private void LabelRegisterProperties()
{

lbl_register.setLayoutX(375);
    lbl_register.setLayoutY(255);
    lbl_register.setTextFill(Color.BLACK);

     // event handler to change color when mouse enters
    EventHandler<MouseEvent> mouseEnterHandler = (MouseEvent event) ->
    {
        lbl_register.setTextFill(Color.RED);
        lbl_register.setCursor(Cursor.HAND);
    };

    lbl_register.setOnMouseEntered(mouseEnterHandler);

    // event handler to change color when mouse exits
    EventHandler<MouseEvent> mouseExitHandler = new EventHandler<MouseEvent>()
    {
        @Override
```

```
            public void handle(MouseEvent event) {
                lbl_register.setTextFill(Color.BLACK);
            }
        };

        lbl_register.setOnMouseExited(mouseExitHandler);

        lbl_register.setOnMouseClicked(onClick ->
        {
            FrmSignUp signUpForm = new FrmSignUp();
            signUpForm.start();
        });
    }

    private void ButtonProperties()
    {
        //Button Properties
        btn_proceed.setAlignment(Pos.CENTER);
        btn_proceed.setDefaultButton(true);
        btn_proceed.setLayoutX(325);
        btn_proceed.setLayoutY(300);
        btn_proceed.setPrefSize(150,30);
        btn_proceed.setStyle("-fx-background-color: #fc5a03");

btn_proceed.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/ClickEffectStyle.css").toExternalForm());
        btn_proceed.setTextFill(Color.BLACK);
        btn_proceed.setCursor(Cursor.HAND);
```

```
        btn_proceed.setOnMouseClicked(event ->
        {
            ExecuteLogin();
        });
    }

    private void ExecuteLogin()
    {

        if(txtUsername.getText().isEmpty() || passwdBox.getText().isEmpty())
            {
                System.out.println("Username or password required!");
                return;
            }
            var user = DatabaseService.Login(txtUsername.getText(), passwdBox.getText());
            if(user != null)
            {
                FrmPatient patient = new FrmPatient(user);
                patient.Show();
            }
    }
}
package thesisuowm.Forms;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import javafx.beans.value.ChangeListener;
```

```
import javafx.beans.value.ObservableValue;
import javafx.geometry.NodeOrientation;
import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextField;
import javafx.scene.control.TextFormatter;
import javafx.scene.control.ToggleGroup;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import javafx.util.StringConverter;
import thesisuowm.Classes.Patient;
import thesisuowm.Classes.User;
import thesisuowm.Database.DatabaseService;

/**
 *
 * @author chala
```

```
 */
public class FrmPatient
{
   private AnchorPane FrmPatient;
   private Label
lbl_title,lbl_smallTitle,lbl_phone,lbl_dob,lbl_genderSelection,lbl_name,lbl_surname,lbl_amka;
   private TextField txt_name,txt_surname,txt_amka,txt_phone;
   private DatePicker dateOfBirth;
   private RadioButton rbt_male,rbt_female;
   private Button btn_submit;
   private String gender;
   private User user;

   public FrmPatient(User user)
   {
      this.FrmPatient = new AnchorPane();
      this.lbl_name = new Label("First name");
      this.lbl_surname = new Label("Surname");
      this.lbl_dob = new Label("Date of Birth");
      this.lbl_phone = new Label("Telephone");
      this.lbl_title = new Label("Medical Diagnostic Tool");
      this.lbl_amka = new Label("Social Security Number");
      this.dateOfBirth = new DatePicker();
      this.lbl_genderSelection = new Label("Gender selection");
      this.lbl_smallTitle = new Label("Finds what makes you sick...");
      this.rbt_male = new RadioButton("Male");
      this.rbt_female = new RadioButton("Female");
      this.btn_submit = new Button("Submit");
```

```
    this.txt_name = new TextField();
    this.txt_surname = new TextField();
    this.txt_amka = new TextField();
    this.txt_phone = new TextField();
    this.gender = "";
    this.user = user;
    SetProperties();
    InitUser(user);
}

private void SetProperties()
{
  FrmPatient.setStyle("-fx-background-color: #6694de");
  SetLabelProperties();
  SetTextFieldProperties();
  SetDatePicker();
  SetRadioButtons();
  SetButtonProperties();
}

private void InitUser(User user)
{

    var exists = DatabaseService.checkIfPatientExists(user.getID());
    if (exists)
    {
        var patient = DatabaseService.getPatientByUserId(user.getID());
```

```
            if (patient != null)
            {
               txt_name.setText(patient.getName());
               txt_surname.setText(patient.getSurname());
               txt_amka.setText(patient.getAMKA());
               txt_phone.setText(patient.getNumber());

dateOfBirth.setValue(LocalDate.parse(patient.getDateOfBirth(),DateTimeFormatter.ofPattern("dd-MM-
yyyy")));
               if(patient.getGender().equals("Male"))
               {
                  rbt_male.setSelected(true);
                  gender = "Male";
               }
               else if(patient.getGender().equals("Female"))
               {
                  rbt_female.setSelected(true);
                  gender = "Female";
               }
            }
         }
         else
         {
            //First time login
            txt_name.setText(user.getName());
            txt_surname.setText(user.getSurname());
         }
      }
```

```
public void Show()
{
    Scene scene = new Scene(this.FrmPatient);
    this.FrmPatient.getChildren().addAll(lbl_name,
            lbl_surname,
            lbl_dob,
            lbl_phone,
            lbl_title,
            lbl_amka,
            dateOfBirth,
            lbl_genderSelection,
            lbl_smallTitle,
            rbt_male,
            rbt_female,
            txt_name,
            txt_surname,
            txt_amka,
            txt_phone,
            btn_submit);

    Stage primaryStage = new Stage();
    primaryStage.setResizable(false);
    primaryStage.setHeight(500);
    primaryStage.setWidth(750);
```

```
        primaryStage.setScene(scene);
        primaryStage.setTitle("Patient Information");
        Image icon = new Image(getClass().getResourceAsStream("/thesisuowm/Icons/patient.png"));
        primaryStage.getIcons().add(icon);
        primaryStage.show();
        this.FrmPatient.requestFocus();
    }

    private void SetLabelProperties()
    {
        //label title
        this.lbl_title.setLayoutX(14);
        this.lbl_title.setLayoutY(14);
        this.lbl_title.setPrefSize(260, 35);
        this.lbl_title.setFont(new Font("Segoe UI Light Italic",24));

        //label small title
        this.lbl_smallTitle.setLayoutX(508);
        this.lbl_smallTitle.setLayoutY(430);
        this.lbl_smallTitle.setPrefSize(208, 9);
        this.lbl_smallTitle.setFont(new Font("Segoe UI Light Italic",18));

        //label name
        this.lbl_name.setLayoutX(154);
        this.lbl_name.setLayoutY(119);
        this.lbl_name.setFont(new Font("System",12));
```

```
//label surname
this.lbl_surname.setLayoutX(456);
this.lbl_surname.setLayoutY(119);
this.lbl_surname.setFont(new Font("System",12));

//label amka
this.lbl_amka.setLayoutX(154);
this.lbl_amka.setLayoutY(198);
this.lbl_amka.setFont(new Font("System",12));

//label dateOfBirth
this.lbl_dob.setLayoutX(154);
this.lbl_dob.setLayoutY(291);
this.lbl_dob.setFont(new Font("System",12));

//label gender
this.lbl_genderSelection.setLayoutX(456);
this.lbl_genderSelection.setLayoutY(291);
this.lbl_genderSelection.setFont(new Font("System",12));

//label phone
this.lbl_phone.setLayoutX(456);
this.lbl_phone.setLayoutY(202);
this.lbl_phone.setFont(new Font("System",12));
```

```
    }

    private void SetDatePicker()
{
    this.dateOfBirth.setLayoutX(154);
    this.dateOfBirth.setLayoutY(311);
    this.dateOfBirth.setNodeOrientation(NodeOrientation.LEFT_TO_RIGHT);
    this.dateOfBirth.setPrefSize(149, 25);

this.dateOfBirth.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/DatePickerStyle.css").toExternalForm());

    // Create a DateTimeFormatter to parse and format dates
    String pattern = "dd-MM-yyyy";
    DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern(pattern);

    // Set the prompt text for the DatePicker
    this.dateOfBirth.setPromptText(pattern.toLowerCase());

    // Create a TextFormatter to auto-insert the '/' separator and format the date
    TextFormatter<LocalDate> textFormatter = new TextFormatter<>(change ->
    {
        String newText = change.getControlNewText();
        if (newText.matches("\\d{0,2}-\\d{0,2}-\\d{0,4}"))
        {
            // If the new text matches the pattern "dd/MM/yyyy", allow the change
            if (newText.length() == 2 || newText.length() == 5)
            {
```

```
            // If the new text contains only day or day and month, add the '/' character
            newText += "/";
        }
        return change;
    } else
    {
        // Otherwise, reject the change
        return null;
    }
});

// Add the TextFormatter to the DatePicker
this.dateOfBirth.setConverter(new StringConverter<LocalDate>() {
    @Override
    public String toString(LocalDate date) {
        return (date != null) ? dateFormatter.format(date) : "";
    }

    @Override
    public LocalDate fromString(String string) {
        return (string != null && !string.isEmpty()) ? LocalDate.parse(string, dateFormatter) : null;
    }
});

this.dateOfBirth.getEditor().setTextFormatter(textFormatter);
}
```

```
    private void SetTextFieldProperties()
    {
        this.txt_name.setLayoutX(154);
        this.txt_name.setLayoutY(140);
        this.txt_name.setPromptText("Only first name");
        this.txt_name.setText(gender);
        this.txt_name.setAlignment(Pos.CENTER);

        this.txt_surname.setLayoutX(456);
        this.txt_surname.setLayoutY(140);
        this.txt_surname.setPromptText("Only last name");
        this.txt_surname.setAlignment(Pos.CENTER);

        this.txt_amka.setLayoutX(154);
        this.txt_amka.setLayoutY(227);
        this.txt_amka.setPromptText("SSNR");
        this.txt_amka.setAlignment(Pos.CENTER);

        // force the field to be numeric only
        this.txt_amka.textProperty().addListener(new ChangeListener<String>()
        {
            public void changed(ObservableValue<? extends String> observable, String oldValue,String
newValue)
            {
                if (!newValue.matches("\\d*"))
                {
                    txt_amka.setText(newValue.replaceAll("[^\\d]", ""));
```

```
            }
        }
    });

    this.txt_phone.setLayoutX(456);
    this.txt_phone.setLayoutY(227);
    this.txt_phone.setPromptText("Telephone");
    this.txt_phone.setAlignment(Pos.CENTER);

    // force the field to be numeric only
    this.txt_phone.textProperty().addListener(new ChangeListener<String>()
    {
        public void changed(ObservableValue<? extends String> observable, String oldValue,String
newValue)
        {
            if (!newValue.matches("\\d*"))
            {
                txt_phone.setText(newValue.replaceAll("[^\\d]", ""));
            }
        }
    });


}

private void SetRadioButtons()
{
    this.rbt_male.setLayoutX(456);
```

```
    this.rbt_male.setLayoutY(315);



    this.rbt_female.setLayoutX(547);
    this.rbt_female.setLayoutY(315);

    ToggleGroup genderGroup = new ToggleGroup();
    this.rbt_female.setToggleGroup(genderGroup);
    this.rbt_male.setToggleGroup(genderGroup);

    this.rbt_male.setOnMouseClicked(event ->
    {
       gender = this.rbt_male.getText();
    });

    this.rbt_female.setOnMouseClicked(event ->
    {
       gender = this.rbt_female.getText();
    });

}

private void SetButtonProperties()
{
    //Button Properties
    this.btn_submit.setAlignment(Pos.CENTER);
    this.btn_submit.setDefaultButton(true);
```

```
        this.btn_submit.setLayoutX(319.0);
        this.btn_submit.setLayoutY(380.0);
        this.btn_submit.setPrefSize(112.0,25.0);
        this.btn_submit.setStyle("-fx-background-color: #fc5a03");

this.btn_submit.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/ClickEffectStyle.css").toExternalForm());
        this.btn_submit.setTextFill(Color.BLACK);
        this.btn_submit.setCursor(Cursor.HAND);
        this.btn_submit.requestFocus();

        this.btn_submit.setOnMouseClicked(event ->
        {
          try
          {
              if (txt_name.getText().isEmpty() ||
              txt_surname.getText().isEmpty() ||
              txt_amka.getText().isEmpty() ||
              txt_phone.getText().isEmpty() ||
              dateOfBirth.getValue() == null)
              {
                 Alert alert = new Alert(AlertType.ERROR);
                 alert.setTitle("Error");
                 alert.setHeaderText("Empty fields detected");
                 alert.setContentText("Please fill in all required fields.");
                 alert.showAndWait();
                 return;
              }
```

```
            var exists = DatabaseService.checkIfPatientExists(user.getID());



            FrmDiagnosis diagnosis = new FrmDiagnosis();
            Patient patient = new Patient(user.getID(),
                                txt_name.getText(),
                                txt_surname.getText(),
                                txt_amka.getText(),
                                gender,
                                dateOfBirth.getValue().format(DateTimeFormatter.ofPattern("dd-MM-
yyyy")),
                                txt_phone.getText());
            if(exists)
            {
               DatabaseService.Insert_Or_Update(patient, exists);
               diagnosis.Show(patient);
            }
            else
            {
               DatabaseService.Insert_Or_Update(patient, exists);
               diagnosis.Show(patient);
            }
        }
        catch (Exception e)
        {
           System.err.println(e.getMessage());
        }
```

```
        });
    }
}
package thesisuowm.Forms;

import javafx.geometry.Pos;
import javafx.scene.Cursor;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Border;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import thesisuowm.Classes.User;
import thesisuowm.Database.DatabaseService;

/**
 *
 * @author chala
 */
public class FrmSignUp
```

```
{
    private TextField txtName;
    private TextField txtSurname;
    private TextField txtUsername;
    private TextField txtEmail;
    private PasswordField txtPassword;
    private PasswordField txtConfirmPassword;
    private Button btnSignUp;

    public void start()
    {
        Stage primaryStage = new Stage();

        primaryStage.setTitle("Sign Up");

        // Create a VBox layout to hold the form elements
        AnchorPane pane = new AnchorPane();
        VBox root = new VBox();
        root.setAlignment(Pos.CENTER);
        root.setStyle("-fx-background-color: #6694de");
        root.setSpacing(15);
        root.setPrefSize(250, 300);
        root.setLayoutX(100);
        root.setLayoutY(15);
        Image icon = new Image(getClass().getResourceAsStream("/thesisuowm/Icons/sign-up.png"));
        primaryStage.getIcons().add(icon);
        pane.setStyle("-fx-background-color: #6694de");
```

```
pane.getChildren().add(root);

// Create the form elements
Label lblTitle = new Label("Create an Account");
lblTitle.setFont(new Font("Segoe UI Light", 20));

txtName = new TextField();
txtName.setPromptText("First Name");

txtSurname = new TextField();
txtSurname.setPromptText("Last Name");

txtUsername = new TextField();
txtUsername.setPromptText("Username");

txtEmail = new TextField();
txtEmail.setPromptText("Email");

txtPassword = new PasswordField();
txtPassword.setPromptText("Password");

txtConfirmPassword = new PasswordField();
txtConfirmPassword.setPromptText("Confirm Password");



btnSignUp = new Button("Sign Up");
```

```
        btnSignUp.setOnAction(e -> signUp());
        btnSignUp.setStyle("-fx-background-color: #fc5a03");

btnSignUp.getStylesheets().add(getClass().getResource("/thesisuowm/Styles/ClickEffectStyle.css").toExte
rnalForm());
        btnSignUp.setTextFill(Color.BLACK);
        btnSignUp.setCursor(Cursor.HAND);

        // Add the form elements to the VBox layout
        root.getChildren().addAll(lblTitle, txtName, txtSurname, txtUsername, txtEmail, txtPassword,
txtConfirmPassword, btnSignUp);

        // Set the scene and show the window
        Scene scene = new Scene(pane, 450, 350);
        primaryStage.setResizable(false);
        primaryStage.setScene(scene);
        primaryStage.show();
        this.btnSignUp.requestFocus();

    }

    private void signUp()
    {
        String name = txtName.getText();
        String surname = txtSurname.getText();
        String username = txtUsername.getText();
        String password = txtPassword.getText();
        String email = txtEmail.getText();
```

```
        User user = new User(name, surname, username, password, email);
        DatabaseService.CreateUser(user);
    }
}
package thesisuowm.KnowledgeBase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 *
 * @author chala
 */
public class RuleEngine
{

    private List<Rule> rules = new ArrayList<>();

    public RuleEngine()
    {
        initializeRules();
    }

    private void initializeRules()
    {
        this.rules.add(new Rule("Covid-19",
```

```
                    Arrays.asList("Cough", "Fever or chills", "Headache", "Loss of taste or
smell","Difficulty breathing","Aches","Diarrhea","Nausea or Vomiting")));

        this.rules.add(new Rule("Common flu",Arrays.asList("Runny or Stuffy nose", "Sore Throat",
"Headache")));

        this.rules.add(new Rule("Pneumonia",Arrays.asList("Cough", "Difficulty breathing", "Fever or
chills", "Chest Pain")));

        this.rules.add(new Rule("Pharyngitis ",Arrays.asList("Fever or
chills","Cough","Rhinorrhea","Hoarseness","Conjunctivitis","Sore Throat","Oral Ulcers")));

        this.rules.add(new Rule("Menigitis",Arrays.asList("Stiff Neck","Fever or
chills","Headache","Photophobia","Confusion")));

        this.rules.add(new Rule("Flu", Arrays.asList("Fever or chills","Cough","Sore Throat","Runny or
Stuffy nose","Aches","Headache","Fatigue")));

    }


    public String execute(List<String> symptoms)

    {

       for (Rule rule : this.rules)

       {

          if (rule.evaluate(symptoms))

          {

             return rule.getDiagnosis();

          }

       }

       return "unrecognized";

    }


    public void addRules(Rule rule)

    {

       this.rules.add(rule);

    }
```

```
}
package thesisuowm.KnowledgeBase;

import java.util.Arrays;
import java.util.List;

/**
 *
 * @author chala
 */
public class Rule
{
   private final  String diagnosis;
   private final  List<String> conditions;

   public Rule(String diagnosis, List<String> conditions)
   {
      this.diagnosis = diagnosis;
      this.conditions = conditions;
   }

   public String getDiagnosis()
   {
      return this.diagnosis;
   }

   public boolean evaluate(List<String> symptoms)
```

```
  {
    var _conditions  = this.conditions.toArray();
    var _symptoms = symptoms.toArray();
    Arrays.sort(_symptoms);
    Arrays.sort(_conditions);

    return Arrays.deepEquals(_conditions, _symptoms);
  }
}
```

SQL Queries

```
CREATE TABLE "patient_records"
(
    "id"       INTEGER,
    "name"  TEXT NOT NULL,
    "surname"         TEXT NOT NULL,
    "condition"       TEXT,
    "diagnosisDate"  TEXT,
    "patientID"       INTEGER,
    PRIMARY KEY("id" AUTOINCREMENT),
    FOREIGN KEY("patientID") REFERENCES "patients"("id")
);
CREATE TABLE "users"
(
    "id"       INTEGER NOT NULL,
    "name"  TEXT,
    "surname"         TEXT,
    "username"       TEXT UNIQUE,
    "password"       TEXT,
```

```
    "email"  TEXT UNIQUE,
    PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE "patients"
(
    "id"       INTEGER,
    "user_id"          INTEGER NOT NULL,
    "name"  TEXT NOT NULL,
    "surname"          TEXT NOT NULL,
    "amka"  TEXT NOT NULL,
    "gender"           TEXT NOT NULL,
    "dob"    TEXT NOT NULL,
    "telephone"        TEXT NOT NULL,
    FOREIGN KEY("user_id") REFERENCES "users"("id"),
    PRIMARY KEY("id" AUTOINCREMENT)
);
```