ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός-δημιουργία εφαρμογής διάγνωσης ψυχικών διαταραχών με τη χρήση της γλώσσας Prolog

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΓΟΓΟΛΟΥ ΑΙΜΙΛΙΑ

ΑΕΜ:2773

Επιβλέπων Καθηγητής: Δόσης Μιχαήλ

Καστοριά 11/2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός-δημιουργία εφαρμογής διάγνωσης ψυχικών διαταραχών με τη χρήση της γλώσσας Prolog

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΓΟΓΟΛΟΥ ΑΙΜΙΛΙΑ

ΑΕΜ:2773

Επιβλέπων : **Δόσης Μιχαήλ**
Ιδιότητα: **Καθηγητής**

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 03/11/2023

Βέργαδος Δημήτριος Δημόκας Νικόλαος Δόσης Μιχαήλ

Καστοριά 11/2023

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κύριο Δόση Μιχαήλ για την συνεργασία, καθώς και για την πολύτιμη συμβουλή του ώστε να ολοκληρωθεί η παρούσα εργασία!

Επίσης τον κύριο Μπάτο Παναγιώτη για την ενθάρρυνση του, τις πολύτιμες συμβουλές και καθοδήγηση του, καθώς και την πίστη του με βοήθησαν στην εκπόνηση της παρούσας εργασίας!

Τέλος ένα μεγάλο Ευχαριστώ στην οικογένεια και τους φίλους μου, για την στήριξη και την ενθάρρυνση τους καθ' όλη την διάρκεια του ταξιδιού αυτού!

Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στο σχεδιασμό και την ανάπτυξη μιας διαγνωστικής εφαρμογής για την ανίχνευση ψυχολογικών ασθενειών με τη χρήση της γλώσσας Prolog και του προγραμματιστικού περιβάλλοντος Visual Prolog 10 IDE. Η διατριβή περιλαμβάνει κεφάλαια σχετικά με τους λειτουργικούς ορισμούς, τα ερευνητικά δεδομένα, τα εργαλεία, την γλώσσα προγραμματισμού. Τα ερευνητικά δεδομένα περιλαμβάνουν δεδομένα σχετικά με την ψυχική διάγνωση, τα ψυχολογικά τεστ στο διαδίκτυο και άλλες εφαρμογές διάγνωσης ψυχολογικών ασθενειών. Η εφαρμογή αναπτύχθηκε με τη χρήση εργαλείων σχεδίασης και διαγραμματικής απεικόνισης UML και ER. Στη συνέχεια, ο κώδικας αναλύθηκε χρησιμοποιώντας δέντρα αποφάσεων και το Visual Prolog. Έπειτα έγινε η παρουσίαση της εφαρμογής η οποία είναι σε θέση να ανιχνεύει ψυχολογικές διαταραχές και να παρέχει διάγνωση με βάση τις απαντήσεις του χρήστη.

Λέξεις Κλειδιά: Διαγνωστικό Πρόγραμμα, Ψυχολογικές διαταραχές, UML, ER, Visual Prolog

Abstract

This thesis focuses on the design and development of a diagnostic application for the detection of psychological diseases using the Prolog language and the Visual Prolog 10 IDE programming environment. The thesis includes chapters on functional definitions, research data, tools, programming language. The research data includes data on mental diagnosis, psychological tests on the web and other applications of psychological illness diagnosis. The application was developed using UML and ER design and diagramming tools. The code was then analyzed using decision trees and Visual Prolog. Then the application was presented which is able to detect psychological disorders and provide diagnosis based on the user's responses.

Keywords: Diagnostic program, Psychological disorders, UML, ER, Visual Prolog

Περιεχόμενα

Λίστα Εικόνων

Εισαγωγή

Στις μέρες μας η τεχνολογία αποτελεί σημαντικό παράγοντα στην καθημερινότητα μας, ιδιαίτερα στην εξέλιξη του τομέα της ιατρικής. Με την χρήση των τεχνολογικών εργαλείων διάγνωσης, ο πάροχος υγειονομικής περίθαλψης κάνει ακριβέστερες διαγνώσεις και εντοπίζει πιο γρήγορα την σωστή χορηγία φαρμακευτικής αγωγής. Στην παρούσα πτυχιακή εργασία, σχεδιάζεται και δημιουργείται μια διαγνωστική εφαρμογή εντοπισμού ψυχολογικών διαταραχών με την χρήση της γλώσσας Visual Prolog.

Στο πρώτο κεφάλαιο αναφέρονται οι λειτουργικοί ορισμοί, τα ερευνητικά δεδομένα και η θεωρία της γλώσσας προγραμματισμού Visual Prolog. Συνεχίζοντας στο δεύτερο κεφάλαιο, αναφερόμαστε στο προγραμματιστικό περιβάλλον Visual Prolog10IDE. Καθώς και στην δημιουργία και ανάλυση των διαγραμμάτων UMLκαι ER. Αναφέροντας τα παραπάνω βοηθάμε τον αναγνώστη, να κατανοήσει την λειτουργικότητα της εφαρμογής καθώς και να μάθει κάποιους βασικούς ορισμούς, ώστε στο επόμενο κεφάλαιο να καταλάβει την δημιουργία και την παρουσίαση της διαγνωστικής εφαρμογής.

Στο τρίτο και τελευταίο κεφάλαιο λοιπόν, πραγματοποιείται η ανάλυση και η παρουσίαση της εφαρμογής. Για την δημιουργία της χρησιμοποιήθηκαν αρκετοί κανόνες, για καλύτερο σχεδιασμό γραφικού περιβάλλοντος και την λειτουργικότητα της.

Η χρήση μιας αντίστοιχης εφαρμογής, θα βοηθούσε τόσο τον πάροχο υγειονομικής περίθαλψης όσο και τον ασθενή.

Κεφάλαιο 1: Λειτουργικοί ορισμοί-Ερευνητικά Δεδομένα

Στο πρώτο κεφάλαιο της παρούσας πτυχιακής εργασίας, θα αναφερθούμε στον ορισμό του διαγνωστικού προγράμματος αλλά και στην χρήση του πάνω στον τομέα της ψυχικής υγείας. Καθώς και στις ψυχολογικές διαταραχές που χρησιμοποιήθηκαν για την υλοποίηση της διαγνωστικής μας εφαρμογής. Στην συνέχεια θα αναφερθούμε στην Prolog ως γλώσσα προγραμματισμού μαζί με κάποιο παράδειγμα και στο τέλος του κεφαλαίου θα αναφερθούμε στις ήδη υπάρχουσες ψυχολογικές διαγνωστικές εφαρμογές που υπάρχουν στην αγορά.

1.1 Διαγνωστικό Πρόγραμμα

Το διαγνωστικό πρόγραμμα αποτελεί μια εφαρμογή λογισμικού, η οποία έχει σχεδιαστεί ώστε να βοηθάει στον εντοπισμό ακόμα και στην επίλυση προβλημάτων χρησιμοποιώντας το υλικό ή το λογισμικό του υπολογιστή. Εκτελούν μια σειρά δοκιμών και αναλύσεων για να εντοπίσουν το πρόβλημα, μπορεί να είναι προ εγκαταστημένα στο σύστημα του υπολογιστή, είτε να μεταφορτωθούν από ιστότοπους τρίτων.

Με την ανάπτυξη που έχει λάβει με την πάροδο του χρόνου, αποτελούν πλέον βασική λειτουργία και μπορεί να το χρησιμοποιήσει κάποιος εύκολα είτε έχει λάβει κάποια ειδική εκπαίδευση είτε όχι. Τα διαγνωστικά προγράμματα όταν πρώτο δημιουργήθηκαν έπρεπε να υπάρχει ισχυρή ένδειξη προβλήματος ώστε να το αναγνωρίσει το λογισμικό και να το αναφέρει.

Πλέον υπάρχουν πολλά και πιο ευέλικτα προγράμματα αυτό ξεκίνησε όταν ήταν συνηθισμένος ένας υπολογιστής στο σπίτι. Υπήρχε σε κάθε υπολογιστή ένα μικρό ενσωματωμένο πρόγραμμα όπως το Checkdisk (CHKISK) στο οποίο ο χρήστης, εκτελούσε βασικές διαγνωστικές εντολές ή πράξεις και αυτό εμφάνιζε αν υπήρχε κάποιο σφάλμα μέσω ειδικών κωδικών σφαλμάτων.Γιανα τις κατανοήσεις έπρεπε να είσαι ειδικά εκπαιδευμένος, ενώ στις μέρεςμας οι κωδικοί αυτοί έχουν αντικατασταθεί με εκθέσεις παραγωγής, οι οποίες είναι πλέον ευανάγνωστες από τους περισσότερους χρήστες.[1]

1.2 Διαγνωστικά Προγράμματα και ψυχική υγεία

Σημαντικό ρόλο παίζει στον τομέα της ιατρικής η χρήση των διαγνωστικών προγραμμάτων, καθώς έχουν δημιουργηθεί ειδικά προγράμματα τα οποία χρησιμοποιούνται ως εργαλεία διαλογής και ταξινόμησης ψυχολογικών ασθενειών. Η πιο σύνηθες μορφή αυτών των διαγνωστικών προγραμμάτων είναι αυτή του ερωτηματολόγιου, το οποίο είναι βασισμένο σε

υπολογιστή. Για την διάγνωση χρησιμοποιούν καθιερωμένα διαγνωστικά κριτήρια όπως το Diagnostic and StatisticalManual of MentalDisorders (DSM) το οποίο θα συμβουλευτεί ένας εξειδικευμένος επαγγελματίας ψυχικής υγείας αφού αξιολογήσει το ιστορικό του ασθενή και τη συνολική κατάσταση της υγείας του.

Εκτός από την παροχή μιας ολοκληρωμένης λίστας όλων των αναγνωρισμένων διαταραχών ψυχικής υγείας, το DSM παραθέτει σημαντικές πληροφορίες:

- Κριτήρια που πρέπει να υπάρχουν για μια ήπια έως σοβαρή διάγνωση
- Κατάλογος γνωστών αιτιών διαταραχών ψυχικής υγείας
- Σχετικές διαφορές φύλου
- Στατιστικά στοιχεία σχετικά με την ηλικία κατά την έναρξη
- Πρόγνωση για διάφορες διαταραχές ψυχικής υγείας
- Έρευνα για τις πιο αποτελεσματικές θεραπευτικές προσεγγίσεις

Είναι πολύ σημαντικό να ξεκαθαρίσουμε πως το διαγνωστικό πρόγραμμα ψυχολογικών ασθενειών είναι διαφορετικό από αυτό του τεστ προσωπικότητας. Οι κλινικοί ιατροί ακολουθούν επίσημα εργαλεία προ συμπτωματικού ελέγχου και αξιολογήσεις, που αναπτύχθηκαν από επιστημονική έρευνα, για να βοηθήσουν στη διάγνωση μιας ψυχικής ασθένειας.[2]

1.3 Ψυχολογικές Ασθένειες

Η Ψυχολογική Ασθένεια πρόκειται για ένα μοτίβο σκέψης ή συμπεριφοράς, η οποία δεν είναι αναπτυξιακά ή κοινωνικά καθορισμένη. Καθορίζεται από τον συνδυασμό του πως αντιλαμβάνεται, ενεργεί, αισθάνεται και σκέφτεται ένα άτομο. Η αναγνώριση και η κατανόηση των ψυχικών διαταραχών, έχει τροποποιηθεί με το πέρασμα των χρόνων και ανά των κόσμο καθώς υπάρχουν ποικίλοι ορισμοί και ταξινομήσεις. Με βάση τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ), πάνω από το ένα τρίτο των ανθρώπων στις περισσότερες χώρες αναφέρονται σε κάποια στιγμή της ζωής τους, προβλήματα τα οποία ταιριάζουν στα καθορισμένα κριτήρια που χρησιμοποιούνται ευρέως.

Οι αιτίες των ψυχολογικών ασθενειών μπορεί να είναι πολλές, συνήθως όμως συνδέονται με τη συνδυασμένη επίδραση γενετικών, βιολογικών, περιβαλλοντολογικών και κοινωνικών παραγόντων. Για την διάγνωση των ασθενειών αυτών υπάρχουν ειδικά σχεδιασμένες ψυχιατρικές κλινικές με ειδικό προσωπικό όπως είναι οι κλινικοί ψυχολόγοι, οι κλινικοί κοινωνικοί λειτουργοί ή τους ψυχίατρους. Όσων αφορά την αντιμετώπιση των ασθενειών αυτών γίνεται μέσω της ψυχοθεραπείας είτε μέσω κάποιας ψυχιατρικής θεραπευτικής αγωγής, πολλές φορές ανάλογα με την ασθένεια γίνεται ένας συνδυασμός και των δύο.

Οι πιο γνωστές ψυχολογικές ασθένειες είναι η κατάθλιψη, η διαταραχή άγχους, η σχιζοφρένεια, η διαταραχή του προστατευτικού παράγοντα(αυτισμός),η διαταραχή παραγωγικών ορμονών(διαταραχής φύλου) κλπ. Στην παρούσα εργασία θα περιοριστούμε στο ψυχολογικό άγχος, στον ψυχαναγκασμό, στην υστερία και στην κατάθλιψη. [3] [3]

1.4 Ψυχολογικό-Άγχος

Το Ψυχολογικό-Άγχος πρόκειται για την διαταραχή η οποία χαρακτηρίζεται από συνεχή ανησυχία και από την αδυναμία του ατόμου, να ελέγξει τα συναισθήματα του. Τα συμπτώματα περιλαμβάνουν ανησυχία για το άγνωστο, υπερένταση, αυξημένη αδρεναλίνη, μυϊκοί σπασμοί, αναστάτωση, συνεχής ανασφάλεια και αντίληψη για πιθανό κίνδυνο καθώς και ακατάσχετες σκέψεις πως κάτι θα συμβεί. Η αιτία της διαταραχής αυτής είναι συνήθως γενετικοί και περιβαλλοντολογικοί παράγοντες, όπως η κληρονομικότητα ή κάποια τραυματική εμπειρία. Για την θεραπεία συνήθως χρησιμοποιούν έναν συνδυασμό από φαρμακευτική αγωγή και ψυχοθεραπείας καθώς και κάποιες αλλαγές στον τρόπο ζωής και της καθημερινής ρουτίνας. Πολλοί από τους ασθενείς για να αντιμετωπίσουν κάποια συμπτώματα τους όπως η κρίση άγχους εκπαιδεύονται από ειδικούς σε τεχνικές αναπνοής.

1.5 Ιδεοψυχαναγκαστική Διαταραχή

Η Ιδεοψυχαναγκαστική διαταραχή ανήκει στις ψυχολογικές διαταραχές που χαρακτηρίζονται από την ύπαρξη αναπόφευκτων, επαναλαμβανόμενων και ανεπιθύμητων σκέψεων αλλά και πράξεων, που προκαλούν σημαντική δυσφορία και αποτελούν εμπόδιο στην καθημερινή ζωή του ασθενούς. Συνήθως οι σκέψεις αυτές είναι συνδεδεμένες με κάποιον φόβο ή ανησυχία όπως ο φόβος για μόλυνση ή από υπερβολικές πράξεις όπως η επαναλαμβανόμενη πλύση των χεριών, μπάνιο πολλές φορές μέσα στην ημέρα και ο έλεγχος για κλειστές πόρτες. Για την θεραπεία χρησιμοποιείται ένας συνδυασμός φαρμακευτικής αγωγής και ψυχοθεραπείας, η οποία βοηθάει στην αντιμετώπιση των φόβων και των σκέψεων του ασθενή.

1.6 Υστερία

Η Υστερία ανήκει στις ψυχογονικές διαταραχές και έχει ως συμπτώματα την απώλεια συνείδησης, την προσωρινή απώλεια μνήμης, παράλυση κα το αίσθημα μίσους. Για την

θεραπεία αυτού του είδους διαταραχής γίνεται ένας συνδυασμός από φαρμακευτική αγωγή και ψυχοθεραπείας η οποία έχει παρατηρηθεί πως τις περισσότερες φορές είναι πιο αποτελεσματική από την φαρμακευτική αγωγή.

1.7 Κατάθλιψη

Η Κατάθλιψη είναι μια διαταραχή της διάθεσης η οποία επηρεάζει το πώς νοιώθει, σκέφτεται και συμπεριφέρεται ο ασθενής. Εμφανίζεται με διαφορετικές μορφές σοβαρότητας, μπορεί να είναι περαστική ή σοβαρή όπου σε αυτή την περίπτωση χρειάζεται ιατρική παρέμβαση είτε μέσω φαρμακευτικής αγωγής είτε μέσω ψυχοθεραπείας. Τα συμπτώματα που εμφανίζει η διαταραχή αυτή είναι απώλεια όρεξης φαγητού, η απώλεια ελπίδας, αίσθημα πίεσης από την ζωή, αίσθημα μίσους και η απώλεια ενδιαφέροντος δραστηριοτήτων οι οποίες παλαιότερα μονοπωλούσαν το ενδιαφέρον του ασθενή.

1.8 Εργαλεία

Με τον όρο εργαλείο ή αλλιώς εργαλείο ανάπτυξης λογισμικού όπως ονομάζεται στον κόσμο της πληροφορικής, αναφερόμαστε σε σχετικά απλά προγράμματα, που μπορούν να συνδυαστούν για να ολοκληρώσουν μια εργασία, χρησιμοποιούνται από προγραμματιστές λογισμικού για τη δημιουργία, τον εντοπισμό σφαλμάτων, τη συντήρηση ή την υποστήριξη άλλων προγραμμάτων και εφαρμογών.[5]

1.9 Prolog

Η Prologείναι μια λογική γλώσσα προγραμματισμού τέταρτης γενιάς, η οποία σχετίζεται με την τεχνητή νοημοσύνη και την υπολογιστική γλωσσολογία. Σε αντίθεση με άλλες γλώσσες προγραμματισμού προορίζεται κυρίως ως δηλωτική γλώσσα, η οποία είναι κατάλληλη για προγράμματα που περιλαμβάνουν συμβολικούς ή μη αριθμητικούς υπολογισμούς. Για αυτό χρησιμοποιείται και στην τεχνητή νοημοσύνη.

Τα τρία κύρια στοιχεία που χρησιμοποιούνται για την εκτέλεση των προγραμμάτων είναι:

- Γεγονότα όπου είναι κατηγόρημα που ισχύει

- Κανόνες όπου είναι εξαφανίσεις γεγονότων που περιέχουν όρους υπό όρους και για την ικανοποίηση ενός κανόνα πρέπει να πληρούνται κάποιες προϋποθέσεις
- Ερωτήσεις όπου μπορούν να απαντηθούν από τα γεγονότα και τους κανόνες

Είναι μια από τις πρώτες γλώσσες προγραμματισμού και ακόμα θεωρείται από τις πιο δημοφιλής με διάφορες δωρεάν και εμπορικές εφαρμογές. Τα πιο σύγχρονα προγραμματιστικά περιβάλλοντα, υποστηρίζουν την δημιουργία γραφικών διεπαφών χρήστη καθώς και δικτυωμένες και διοικητικές εφαρμογές.

Η γλώσσα Prologσυναντάται σε διάφορους επιστημονικούς τομείς όπως:

- Ευφυής Ανάκτηση Βάσεων Δεδομένων
- Κατανόηση Φυσικής Αγωγής
- Γλώσσα Προδιαγραφών
- Μηχανική Μάθηση
- Σχεδιασμός Ρομπότ
- Επίλυση Προβλημάτων
- Συστήματα Αυτοματισμού

Ο βασικός τρόπος εκτέλεσης των προγραμμάτων είναι μέσω των ερωτημάτων εφόσον πρόκειται για μια διερμηνευόμενη γλώσσα.[6][7]

Όπως θα δούμε και στο παράδειγμα γενεαλογικού δέντρου μέσω του λογισμικού Visual Prolog που παρατίθεται παρακάτω.

TaskWindow.pro (TaskWindow\)

```
1  % Licensed under the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).
2
3  implement taskWindow inherits applicationWindow
4     open vpiDomains
5
6  facts
7     currentDirectory : string := "".
8
9  domains
10    gender = female; male.
11
12 class facts - familyDB
13    person : (string Name, gender Gender).
14    parent : (string Person, string Parent).
15
16 class predicates
17    father : (string Person, string Father) nondeterm anyflow.
18 clauses
19    father(Person, Father) :-
20       parent(Person, Father),
21       person(Father, male()).
22
23 class predicates
24    grandFather : (string Person [out], string Grandfather [out]) nondeterm.
25 clauses
26    grandfather(Person, Grandfather) :-
27       parent(Person, Parent),
28       father(Parent, Grandfather).
29
30 class predicates
31    ancestor : (string Person, string Ancestor [out]) nondeterm.
32 clauses
33    ancestor(Person, Ancestor) :-
34       parent(Person, Ancestor).
35    ancestor(Person, Ancestor) :-
36       parent(Person, P1),
37       ancestor(P1, Ancestor).
38
39 class predicates
40    reconsult : (string FileName).
41 clauses
42    reconsult(Filename) :-
43       retractFactDB(familyDB),
44       file::consult(Filename, familyDB).
45
46 constants
47    mdiProperty : boolean = true.
48
49 clauses
```

1:1 Insert

Εικόνα 1 Παράδειγμα Visual Prolog 1

TaskWindow.pro (TaskWindow\)

```
85  clauses
86     onFileOpen(_Source, _MenuTag) :-
87        currentDirectory := directory::getCurrentDirectory(),
88        directory::setCurrentDirectory(@"..\"),
89        Filename =
90           vpiCommonDialogs::getFileName("*.txt", ["Family data files (*.txt)", "*.txt", "All files", "*.*"], "Load family database", [], ".", _),
91        directory::setCurrentDirectory(currentDirectory),
92        !,
93        reconsult(Filename),
94        stdIO::writef("Database % loaded\n", Filename).
95
96     onFileOpen(_Source, _MenuTag) :-
97        directory::setCurrentDirectory(currentDirectory).
98
99  class predicates
100    onQueryFather : window::menuItemListener.
101 clauses
102    onQueryFather(_Source, _MenuTag) :-
103       stdIO::write("\nfather test\n"),
104       father(X, Y),
105       stdIO::writef("% is the father of %\n", Y, X),
106       fail.
107
108    onQueryFather(_Source, _MenuTag).
109
110 class predicates
111    onQueryGrandfather : window::menuItemListener.
112 clauses
113    onQueryGrandFather(_Source, _MenuTag) :-
114       stdIO::write("\ngrandFather test\n"),
115       grandfather(X, Y),
116       stdIO::writef("% is the grandfather of %\n", Y, X),
117       fail.
118
119    onQueryGrandfather(_Source, _MenuTag).
120
121 predicates
122    onQueryAncestorOf : window::menuItemListener.
123 clauses
124    onQueryAncestorOf(_Source, _MenuTag) :-
125       X = ancestorDialog::tryGetName(This),
126       stdIO::writef("\nancestor of % test\n", X),
127       ancestor(X, Y),
128       stdIO::writef("% is the ancestor of %\n", Y, X),
129       fail.
130
131    onQueryAncestorOf(_Source, _MenuTag).
132
133 % This code is maintained automatically, do not update it manually. 16:58:59-22.6.2006
```

1:1 Insert

Εικόνα 2 Παράδειγμα Visual Prolog 2

Το παραπάνω παράδειγμα γενεαλογικού δένδρου, βρίσκεται στον φάκελο με τα παραδείγματα που εγκαθιστάτε μαζί με το λειτουργικό περιβάλλον της Visual Prolog και πρόκειται για το family2.

Όπως είναι στην Prologένα γενεαλογικό δένδρο έτσι και εδώ μας επιτρέπει να κάνουμε ερωτήσεις σχετικά με τις σχέσεις μεταξύ των διάφορων ατόμων που βρίσκονται στο γενεαλογικό δένδρο. Η έξοδος θα εμφανίζεται στο παράθυρο της κονσόλας.

Παρατηρούμε πως στην Visual Prologκαθορίζουμε τα πεδία και τα κατηγορήματα, πριν τον ορισμό των ρητρών. Επίσης για την εκτέλεση ερωτημάτων κατά την εκτέλεση του προγράμματος, πρέπει να προστεθεί πρώτα ένα τμήμα στόχου.

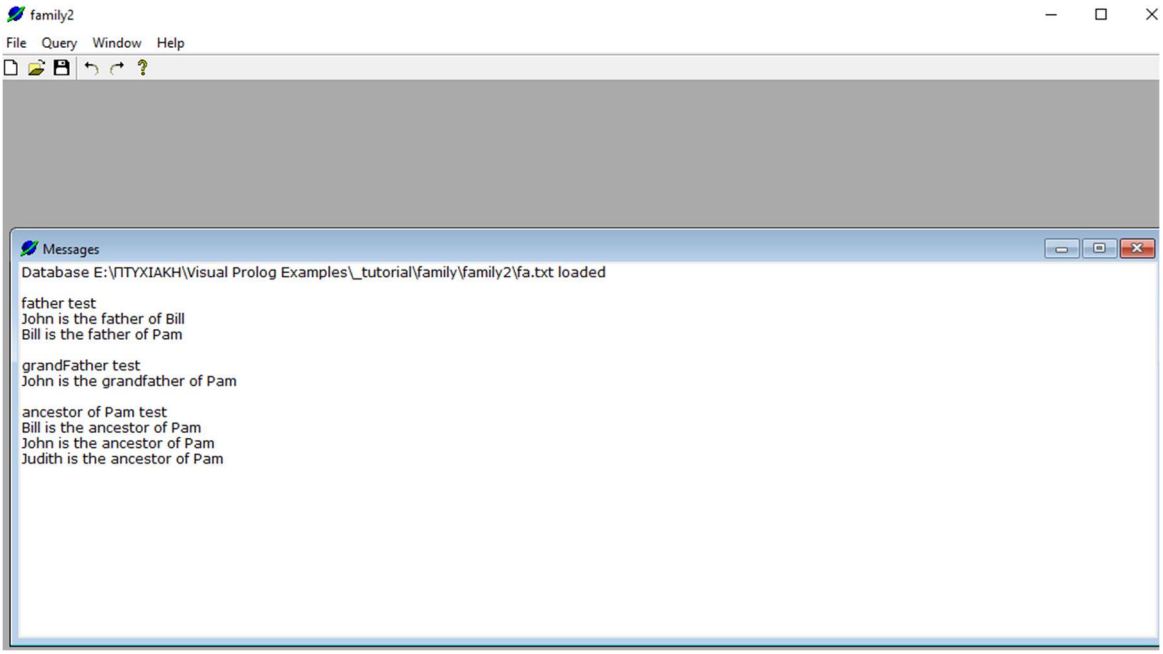
Αφού ορίσουμε τα πεδία και τα κατηγορήματα, ορίζουμε τον τομέα με όνομα “person” ως συμβολοσειρά για την αναπαράσταση των ονομάτων των ατόμων στο γενεαλογικό δένδρο. Συνεχίζουμε ορίζοντας διάφορα κατηγορήματα για τους διάφορους τύπους σχέσεων και το φύλλο.

Στην συνέχεια ορίζουμε τις ρήτρες για το γενεαλογικό δένδρο, το οποίο περιλαμβάνει τις σχέσεις των γονέων, το φύλλο κάθε ατόμου και τους κανόνες για τους διαφορετικούς τύπους σχέσεων. Οι κανόνες ορίζονται με την χρήση του “:-”, που σημαίνει “αν”.

Παραδείγματος χάρη ο κανόνας του παππού όπως και 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 γενεαλογικό δένδρο. 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.



Εικόνα 3 Αποτελέσματα παραδείγματος Visual Prolog

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").
```

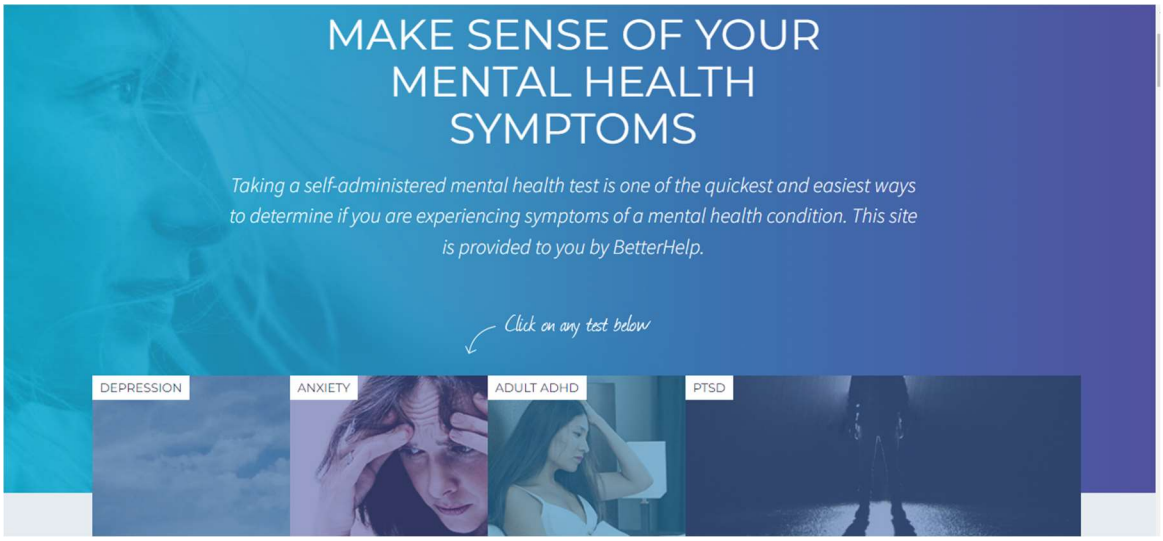
1.10 Ερευνητικά Δεδομένα

Στην παρούσα ενότητα θα αναφερθούμε σε τρείς διαγνωστικές εφαρμογές ψυχικής υγείας. Όπου στόχος τους είναι να διευκολύνουν τον ασθενή για μια εύκολη διάγνωση. Οι δύο πρώτες

αναφορές που γίνονται, είναι για τεστ αξιολόγησης ενώ η τρίτη πρόκειται για μια διαδικτυακή εφαρμογή στην οποία μπορείς να κλείσεις κάποια συνεδρία.

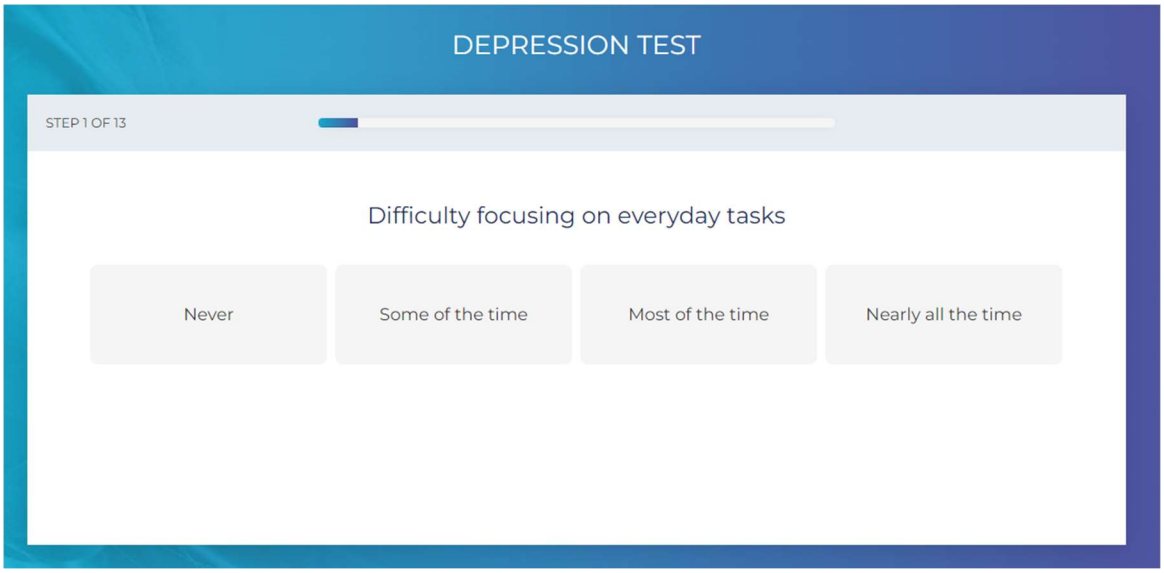
Είναι σημαντικό να σημειωθεί ότι, ενώ αυτά τα εργαλεία μπορούν να είναι χρήσιμα στη διαγνωστική διαδικασία, δεν θα πρέπει να αντικαταστήσουν μια ολοκληρωμένη αξιολόγηση από εξειδικευμένο επαγγελματία ψυχικής υγείας. Αυτές οι εφαρμογές έχουν σχεδιαστεί για να υποστηρίζουν την κλινική κρίση και να παρέχουν πρόσθετες πληροφορίες, αλλά η τεχνογνωσία και η ερμηνεία ενός εκπαιδευμένου επαγγελματία είναι ζωτικής σημασίας για την ακριβή διάγνωση και την κατάλληλη θεραπεία.

1.11 Mind Diagnostics



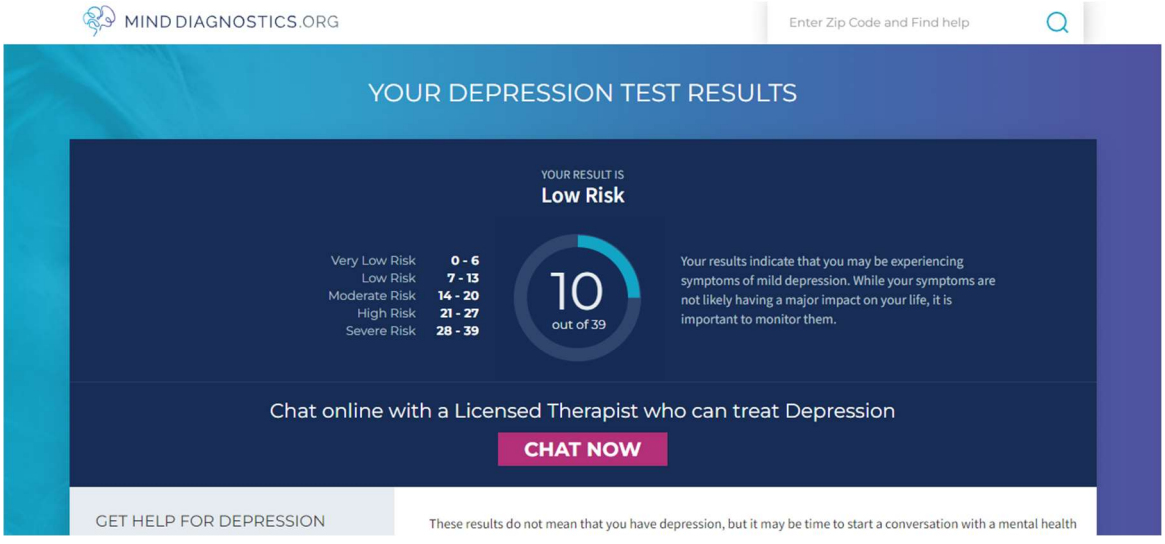
Εικόνα4 Mind Diagnostic Home Page

ToMind Diagnostic είναι μια διαδικτυακή σελίδα αξιολόγησης ψυχικής υγείας. Πρόκειται για μια υπηρεσία που παρέχεται από την Better Help, όπου ο χρήστης μπορεί είτε να μιλήσει με κάποιον θεραπευτή, είτε να προβεί σε κάποιο δωρεάν τεστ αυτό-αξιολόγησης που παρέχει η σελίδα.[8]



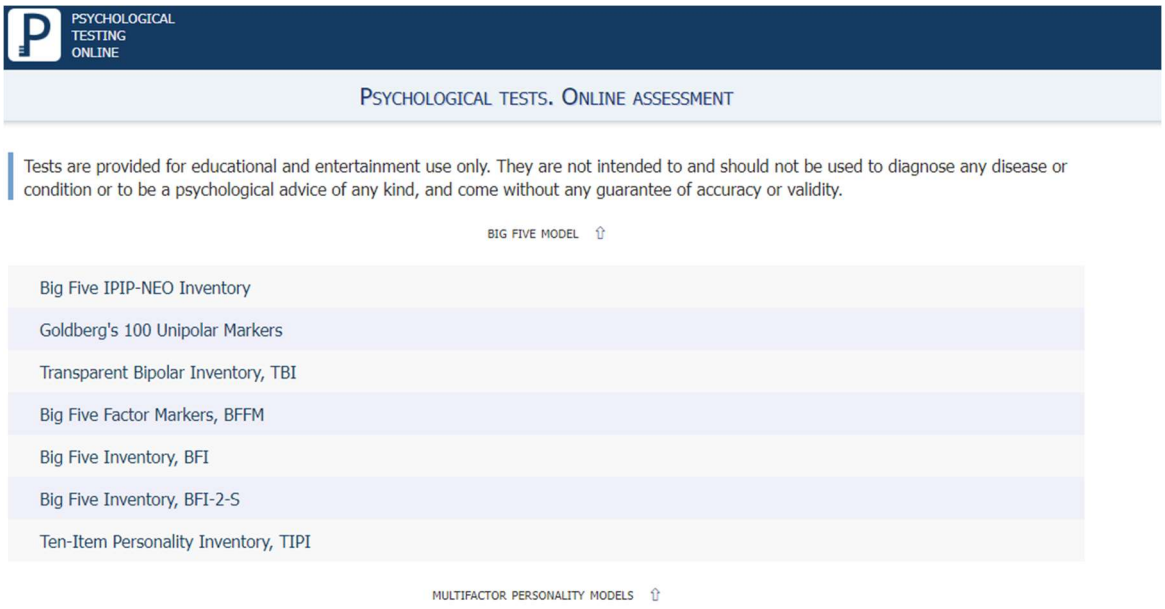
Εικόνα 4 Mind Diagnostics Ερώτηση

Τα τεστ αυτό-αξιολόγησης έχουν την μορφή ερωτηματολόγιου, ο χρήστης απαντάει κατά πόσο συχνά ή όχι αισθάνεται το σύμπτωμα της ερώτησης. Ανάλογα με τις απαντήσεις του χρήστη το σύστημα εμφανίζει το ποσοστό της πιθανότητας να πάσχει από αυτή την ασθένεια και του δίνει την επιλογή να έρθει σε επαφή με κάποιον ειδικευμένο θεραπευτή.



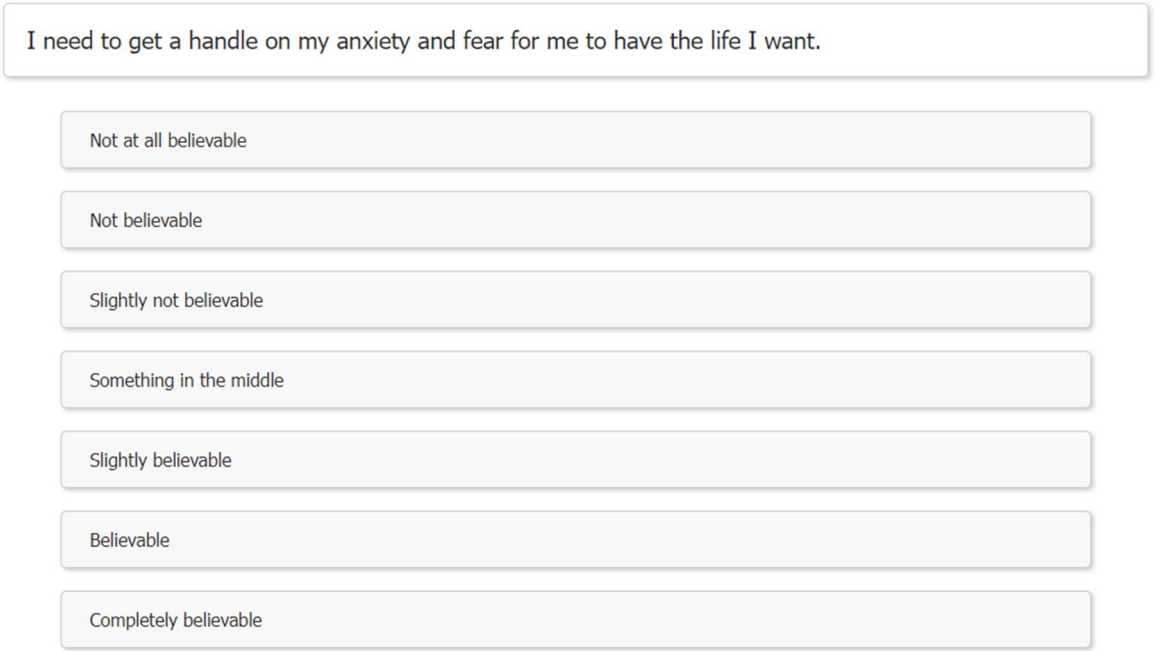
Εικόνα 5 Mind Diagnostics Αποτελέσματα

1.12 Psychological Testing Online



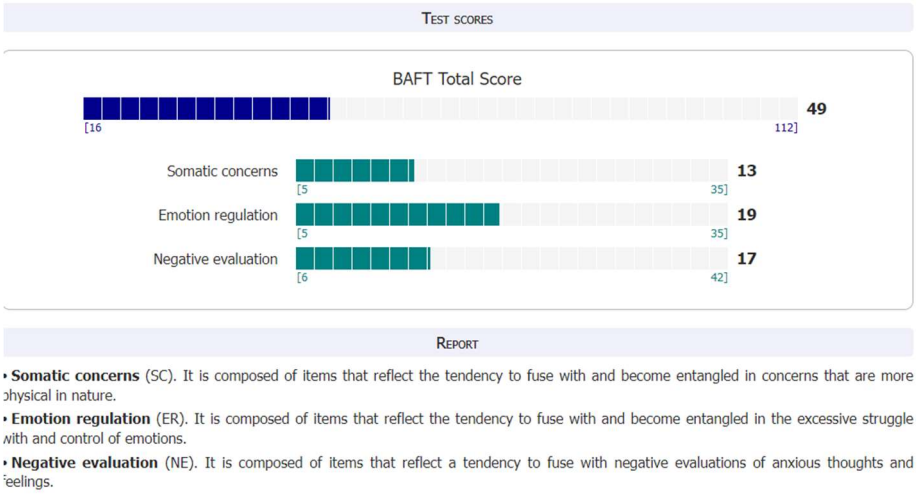
Εικόνα6 Psychological Testing Online Home Page

Το Psychological Testing Online είναι ένας δωρεάν προς χρήση ιστότοπος αφιερωμένος στην παροχή πρόσβασης σε επαγγελματίες ψυχολογίας, φοιτητές και στο ευρύ κοινό σε ακαδημαϊκά επικυρωμένα εργαλεία ψυχολογικής αξιολόγησης με απλή διεπαφή και αυτοματοποιημένη βαθμολόγηση.[9]



Εικόνα 8 Psychological Testing OnlineΕρωτήσεις

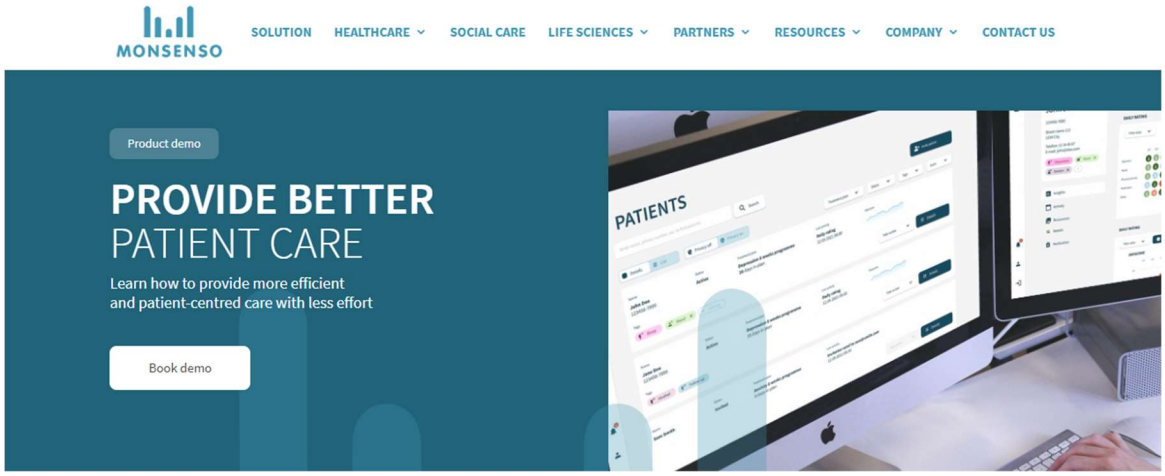
Ο χρήστης επιλέγει το τεστ αυτό-αξιολόγησης που τον ενδιαφέρει, σε ορισμένα ζητάει και κάποια προσωπικά του στοιχεία. Όπως και η προηγούμενη ιστοσελίδα αυτό-αξιολόγησης έτσι και αυτή χρησιμοποιεί την μορφή ερωτηματολογίου στις ερωτήσεις της. Εφόσον ολοκληρώσει το τεστ, το σύστημα θα του εμφανίσει το ποσοστό πιθανότητας να πάσχει από την ασθένεια του τεστ που επέλεξε να εκτελέσει.



Εικόνα 9 Psychological Testing OnlineΑποτελέσματα

Τα τεστ του παραπάνω ιστότοπου, δεν προορίζονται και δεν πρέπει να χρησιμοποιούνται για τη διάγνωση οποιασδήποτε ασθένειας ή πάθησης ή για ψυχολογική συμβουλή οποιουδήποτε είδους και παρέχονται χωρίς καμία εγγύηση ακρίβειας ή εγκυρότητας.

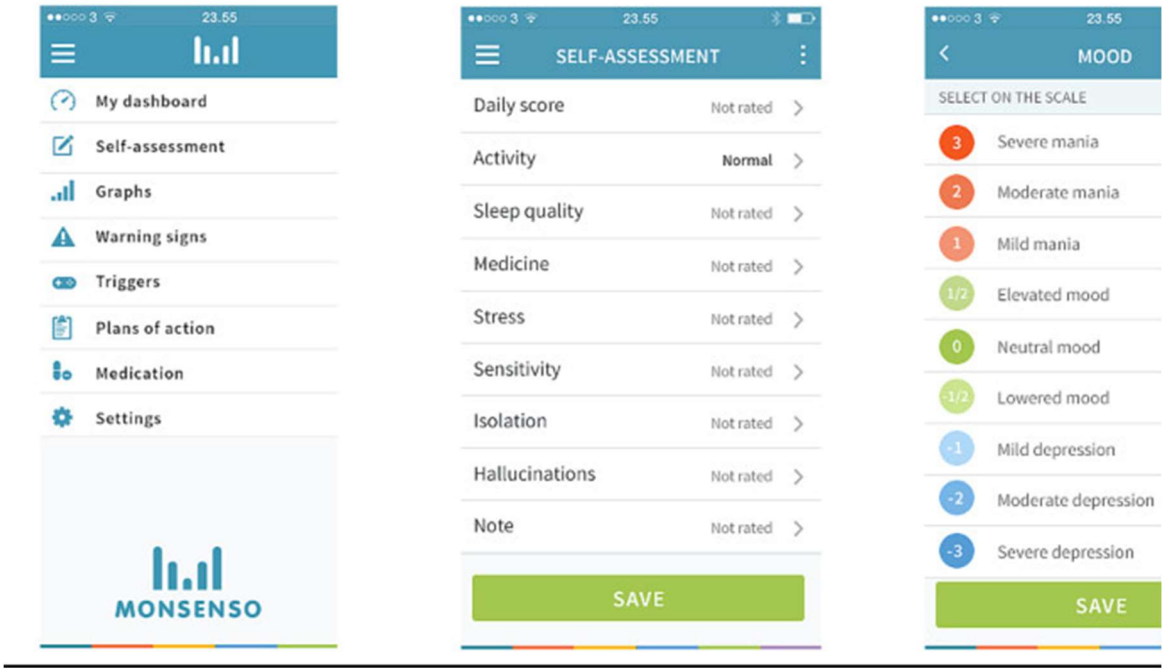
1.13 Monsenso



Εικόνα 10 Monsenso HomePage

Η Monsenso είναι μια εταιρεία ψηφιακής υγείας που ειδικεύεται στην παροχή λύσεων υγείας μέσω κινητών τηλεφώνων για την ψυχική υγεία και ευεξία. Στόχος της είναι η ενδυνάμωση των ατόμων, των φορέων υγειονομικής περίθαλψης και την συμβολή παροχής καλύτερης ψυχικής υγείας με χαμηλότερο κόστος μέσω της διαδικτυακής πλατφόρμας. Το πρόγραμμα χρησιμοποιεί την καθοδηγούμενη από τα δεδομένα νοημοσύνη στη διάγνωση, την πρόληψη και τη θεραπεία.

Πρόκειται για μια ολοκληρωμένη πλατφόρμα κινητής υγείας που έχει σχεδιαστεί για να βοηθά τα άτομα, τους επαγγελματίες υγείας και τους ερευνητές στη διαχείριση των καταστάσεων ψυχικής υγείας. Η πλατφόρμα συνδυάζει εφαρμογές smartphone, wearables και αναλύσεις που βασίζονται στο cloud για τη συλλογή και ανάλυση δεδομένων που σχετίζονται με τα συμπτώματα ψυχικής υγείας των χρηστών, τις συμπεριφορές και τους περιβαλλοντικούς παράγοντες. Τα δεδομένα αυτά μπορούν στη συνέχεια να αξιοποιηθούν για την παροχή εξατομικευμένων πληροφοριών, την παρακολούθηση της προόδου και την προσφορά παρεμβάσεων ή συστάσεων θεραπείας.



Εικόνα 7 Monsenso App

Η εφαρμογή επιτρέπει στους χρήστες να παρακολουθούν τη διάθεση, τις συνήθειες ύπνου, τη σωματική δραστηριότητα και την τήρηση της φαρμακευτικής αγωγής, μεταξύ άλλων παραμέτρων. Προσφέρει επίσης εργαλεία αυτοβοήθειας, όπως ασκήσεις γνωσιακής-συμπεριφορικής θεραπείας και τεχνικές ενσυνειδητότητας, για να ενδυναμώσει τα άτομα στο ταξίδι τους στην ψυχική υγεία. Η πλατφόρμα της Monsenso έχει σχεδιαστεί για να διευκολύνει

την επικοινωνία και τη συνεργασία μεταξύ των ασθενών και των φορέων παροχής υγειονομικής περίθαλψης. Επιτρέπει την ασφαλή ανταλλαγή δεδομένων και την παρακολούθηση σε πραγματικό χρόνο, επιτρέποντας στους επαγγελματίες υγείας να αξιολογούν εξ αποστάσεως την ευημερία των ασθενών τους, να προσαρμόζουν τα σχέδια θεραπείας και να παρέχουν έγκαιρη υποστήριξη. Οι δυνατότητες ανάλυσης δεδομένων της πλατφόρμας συμβάλλουν επίσης στις ερευνητικές προσπάθειες και μπορούν να βοηθήσουν στον εντοπισμό μοτίβων, παραγόντων κινδύνου και πιθανών θεραπευτικών αποτελεσμάτων.[10]

Κεφάλαιο 2:Σχεδιασμός Εφαρμογής

Στο δεύτερο κεφάλαιο θα δούμε τα εργαλεία σχεδιασμού που χρειάζεται ένας προγραμματιστής μαζί με μερικά παραδείγματα, θα αναλύσουμε τα εργαλεία που χρησιμοποιήθηκαν για την εκπόνηση της εργασίας αυτής. Κλείνοντας θα αναλύσουμε τα διαγράμματα UML και ER.

2.1 Εργαλεία Σχεδιασμού

Τα πιο βασικά εργαλεία για την υλοποίηση μιας εφαρμογής λογισμικού είναι:

- ένας συντάκτης κειμένων (editor) με τον οποίο και γράφει το αρχικό πρόγραμμα, που ονομάζεται πηγαίο πρόγραμμα ή κώδικας (sourcecode).
- ένα μεταφραστικό πρόγραμμα (μεταγλωττιστή ή διερμηνευτή), το οποίο μεταφράζει το πηγαίο πρόγραμμα σε αντικείμενο πρόγραμμα ή κώδικα (objectcode). Το μεταφραστικό πρόγραμμα ελέγχει το πηγαίο πρόγραμμα για συντακτικά λάθη, εμφανίζει κατάλληλα διαγνωστικά μηνύματα, εάν βρεθούν λάθη, και μόνο αν δεν υπάρχουν λάθη παράγεται το αντικείμενο πρόγραμμα. Το αντικείμενο πρόγραμμα είναι σε γλώσσα μηχανής, αλλά δεν είναι ακόμη εκτελέσιμο από τον υπολογιστή και πρέπει να περάσει από κάποιες άλλες διαδικασίες.
- ένα ειδικό πρόγραμμα που ονομάζεται συνδέτης (linker), το οποίο πολλές φορές συνδέει το αντικείμενο πρόγραμμα ή ένα σύνολο από αντικείμενα προγράμματα με έτοιμα υπό προγράμματα της βιβλιοθήκης της γλώσσας προγραμματισμού ή του προγραμματιστή. Το τελικό πρόγραμμα που παράγεται είναι το εκτελέσιμο πρόγραμμα ή κώδικας (executablecode), είναι διατυπωμένο σε γλώσσα μηχανής και μπορεί να εκτελεστεί άμεσα από τον επεξεργαστή του υπολογιστή.
- εργαλεία εντοπισμού λαθών (debuggers) με τα οποία ο προγραμματιστής παρακολουθεί τι ακριβώς συμβαίνει στο παρασκήνιο κατά την εκτέλεση ενός προγράμματος.

Οι επαγγελματίες προγραμματιστές χρησιμοποιούν για την σχεδίαση, την κωδικοποίηση, τον έλεγχο λαθών και τη συντήρηση μιας εφαρμογής ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment -IDE) συνδυάζουν τα χαρακτηριστικά πολλών εργαλείων σε ένα πακέτο.

Μερικά από τα IDE είναι:

- Visual Studio
- Eclipse
- Atom
- NetBeans
- PyCharm
- Intellij IDEA
- Code::Blocks
- Aptana Studio 3
- Komodo IDE
- RubyMine

Αυτά είναι τα δέκα πιο γνωστά IDE προγράμματα που χρησιμοποιεί ένας προγραμματιστής για την ανάπτυξη εφαρμογών σε εταιρικό επίπεδο.[11][12]

2.2 Visual Prolog

Η Visual Prolog είναι μια γλώσσα προγραμματισμού και ένα ολοκληρωμένο περιβάλλον ανάπτυξης IDE,το οποίο χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών βασισμένων στην λογική και δηλωτικών εφαρμογών.

Πρόκειται για μια γλώσσα πολλαπλών παραδειγμάτων, βασισμένη στην γλώσσα Prolog επεκτείνοντας την με δυνατότητες αντικειμενοστραφούς προγραμματισμού (OOP). Θεωρείται μια ισχυρή και ασφαλής γλώσσα υψηλού επιπέδου, η οποία παρέχει έναν συνδυασμό λογικού προγραμματισμού και παραδειγμάτων αντικειμενοστραφούς προγραμματισμού. Ακολουθεί το πρότυπο λογικού προγραμματισμού, το οποίο βασίζεται σε ένα επίσημο σύστημα συμβολικής λογικής. Υποστηρίζει διάφορους τύπους δεδομένων, αλλά μας δίνει την δυνατότητα να ορίσουμε τους δικούς μας τύπους δεδομένων, με την χρήση των δομών και των κλάσεων καθώς και να ορίσουμε γεγονότα και κανόνες, ώστε να αναπαραστήσουμε σχέσεις και να εκτελέσουμε λογικά συμπεράσματα.

Η Visual Prolog είναι ένα σύστημα στατικού τύπου, όπου επιβάλει τον έλεγχο τύπου στην στιγμή της μεταγλώττισης. Χρησιμοποιείται ευρέως στον ακαδημαϊκό χώρο και τη βιομηχανία για εργασίες όπως η τεχνητή νοημοσύνη, τα έμπειρα συστήματα, η επεξεργασία φυσικής γλώσσας και ο συμβολικός υπολογισμός. Καθώς διαθέτει ένα πλούσιο σύνολο δυνατοτήτων και εργαλείων τα οποία υποστηρίζουν την αποτελεσματική ανάπτυξη λογισμικού και την λογική επίλυση προβλημάτων.

Το ολοκληρωμένο IDEπου μας παρέχει, μας δίνει διάφορες δυνατότητες όπως πρόγραμμα επεξεργασίας κώδικα, διαχείριση έργου, μεταγλωττιστή, εντοπισμό σφαλμάτων και εργαλεία σχεδίασης GUI. Το IDEπροσφέρει μια φιλική προς τον χρήστη διεπαφή και διευκολύνει την διαδικασία ανάπτυξης. Επίσης είναι ένα από τα πιο σύγχρονα προγραμματιστικά περιβάλλοντα, οπότε μπορούμε να δημιουργήσομε γραφικές διεπαφές χρήστη καθώς και διοικητικές εφαρμογές.

Η Visual Prolog μπορεί να μεταγλωττιστεί και να εκτελεστεί σε διάφορες πλατφόρμες συμπεριλαμβανομένων των Windows, Linux και MacOs. Προσφέρει διαλειτουργικότητα με άλλες γλώσσες προγραμματισμού, μέσω μηχανισμών όπως διεπαφές ξένων γλωσσών επιτρέποντας μας, να ενσωματώσουμε κώδικα Visual Prologμε κώδικα γραμμένο σε διαφορετικές γλώσσες προγραμματισμού.

Παρακάτω θα αναφερθούμε σε κάποιες λέξεις-κλειδιά για το περιβάλλον ανάπτυξης της Visual Prolog.

domains: Οι τομείς στο Visual Prolog αναφέρονται στους τύπους δεδομένων ή τις δομές δεδομένων που χρησιμοποιούνται για τον καθορισμό του εύρους των πιθανών τιμών για τις μεταβλητές. Το Visual Prolog παρέχει διάφορους ενσωματωμένους τομείς όπως ακέραιος, πραγματικός, char, συμβολοσειρά κ.λπ. Μπορείτε επίσης να ορίσετε τους προσαρμοσμένους τομείς σας χρησιμοποιώντας τη δήλωση τομέα.

facts: Στο Visual Prolog, τα γεγονότα αντιπροσωπεύουν τις βεβαιωμένες ή γνωστές πληροφορίες σχετικά με τον τομέα του προβλήματος. Χρησιμοποιούνται για να δηλώσουν σχέσεις ή ιδιότητες που ισχύουν στο πρόγραμμα. Τα γεγονότα τυπικά αναπαριστώνται ως μια συλλογή κατηγορηματικών όρων που παρέχουν πληροφορίες για αντικείμενα ή οντότητες.

classfacts: Στο Visual Prolog, τα γεγονότα κλάσης είναι παρόμοια με τα κανονικά γεγονότα, αλλά σχετίζονται με αντικειμενοστρεφείς έννοιες προγραμματισμού. Αντιπροσωπεύουν τις ιδιότητες ή τις ιδιότητες των στιγμιότυπων (αντικειμένων) μιας κλάσης. Τα γεγονότα κλάσης ορίζονται μέσα σε μια κλάση και μπορούν να χρησιμοποιηθούν για την προετοιμασία των ιδιοτήτων των αντικειμένων.

predicates: Τα κατηγορήματα στο Visual Prolog ορίζουν λογικές σχέσεις ή συνθήκες. Μπορούν να χρησιμοποιηθούν για την έκφραση κανόνων, υπολογισμών ή ερωτημάτων. Τα κατηγορήματα μπορούν να λάβουν ορίσματα και μπορεί να είναι είτε αληθή είτε ψευδή. Ορίζονται χρησιμοποιώντας τη λέξη-κλειδί κατηγόρημα.

clauses: Οι όροι στο Visual Prolog περιέχουν τις λεπτομέρειες υλοποίησης των κατηγορημάτων. Καθορίζουν τους λογικούς κανόνες ή προϋποθέσεις που πρέπει να

πληρούνται για να είναι αληθές το κατηγόρημα. Οι όροι αποτελούνται από ένα κεφάλι και ένα σώμα, όπου το κεφάλι αντιπροσωπεύει τον στόχο ή το συμπέρασμα και το σώμα περιέχει τις προϋποθέσεις ή τους υποστόχους που πρέπει να ικανοποιηθούν.

class: Στο Visual Prolog, μια κλάση είναι ένα σχέδιο ή πρότυπο για τη δημιουργία αντικειμένων. Καθορίζει τις ιδιότητες (χαρακτηριστικά) και τις συμπεριφορές (μέθοδοι) που διαθέτουν τα αντικείμενα αυτής της κλάσης. Οι κλάσεις παρέχουν έναν τρόπο οργάνωσης και δομής κώδικα χρησιμοποιώντας αντικειμενοστρεφείς αρχές προγραμματισμού, όπως η ενθυλάκωση, η κληρονομικότητα και ο πολυμορφισμός.

implement/ endimplement: Στο Visual Prolog, οι λέξεις-κλειδιά "υλοποίηση" και "τελική υλοποίηση" χρησιμοποιούνται για τον ορισμό της ενότητας υλοποίησης μιας κλάσης. Σε αυτήν την ενότητα καθορίζετε τις ιδιότητες και τις μεθόδους της κλάσης. Η ενότητα "υλοποίηση" τοποθετείται συνήθως μετά τη δήλωση κλάσης και πριν από τη δήλωση "τελική κλάση".

constants: Οι σταθερές στο Visual Prolog είναι σταθερές τιμές που δεν αλλάζουν κατά την εκτέλεση του προγράμματος. Χρησιμοποιούνται για την αναπαράσταση σταθερών ή αμετάβλητων δεδομένων που παραμένουν σταθερά σε όλο το πρόγραμμα. Οι σταθερές μπορεί να είναι διαφόρων τύπων δεδομένων, όπως ακέραιοι, πραγματικοί αριθμοί, χαρακτήρες ή συμβολοσειρές. Συνήθως δηλώνονται χρησιμοποιώντας τη λέξη-κλειδί const.

open: Στο Visual Prolog, η λέξη-κλειδί "ανοιχτό" χρησιμοποιείται για την εισαγωγή μιας λειτουργικής μονάδας ή μιας βιβλιοθήκης στο τρέχον πρόγραμμα. Σας επιτρέπει να έχετε πρόσβαση στα κατηγορήματα, τις κλάσεις ή άλλες οντότητες που ορίζονται στην εισαγόμενη λειτουργική μονάδα. Η δήλωση "ανοιχτή" τοποθετείται συνήθως στην αρχή του προγράμματος ή στην κορυφή ενός αρχείου για να καταστήσει τις εισαγόμενες οντότητες διαθέσιμες προς χρήση.[13][14]

2.3 Διαγράμματα UML και ER

Για την δημιουργία της διαγνωστικής μας εφαρμογής χρησιμοποιήθηκαν UMLκαι ER(Entity-relations) διαγράμματα, τα οποία είναι διαγράμματα αναπαράστασης δομής τους συστήματος και διάγραμμα αναπαράστασης δεδομένων.

Συγκεκριμένα η UMLθεωρείται μια δυναμική γλώσσα στην οποία, μπορείς να δημιουργείς-χρησιμοποιείς την δική σου διάλεκτο ώστε να περιγράψεις ένα σύστημα με διαφορετικούς

τρόπους και από διαφορετικές οπτικές γωνίες(views)χωρίς απαραίτητα να ορίσεις πια διαδικασία θα ακολουθήσουμε.

Μερικά από τα διαγράμματα μπορεί να είναι ισοδύναμα, χωρίζονται όμως σε δύο κατηγορίες περιγραφής οι οποίες είναι, η στατική δομή και η δυναμική δομή.

Η στατική δομή αποτελείται από

- Διαγράμματα κλάσεων(class diagram)
- Διαγράμματα αντικειμένων(object diagram)
- Διαγράμματα στοιχείων(component diagram)
- Διαγράμματα διάταξης(deployment diagram)

Ενώ τα διαγράμματα δυναμικής δομής αποτελούνται από

- Διαγράμματα περιπτώσεων χρήσης(usecasediagram)
- Διαγράμματα δραστηριοτήτων(activity diagram)
- Διαγράμματα ακολουθίας(sequence diagram)
- Διαγράμματα συνεργασίας(collaboration diagram)
- Διαγράμματα καταστάσεων(statechart diagram)

Για την δημιουργία της εφαρμογής μας χρησιμοποιήσαμε διάγραμμα δυναμικής δομής και συγκεκριμένα το διάγραμμα δραστηριοτήτων(activity diagram). Το διάγραμμα δραστηριοτήτων περιγράφει την ροή των εργασιών μέσα στο σύστημα. Χρησιμοποιείτε για την περιγραφή μιας περίπτωσης χρήσης, την περιγραφή των ροών εργασίας(workflow), για τις διεργασίες(processes) και των επιχειρηματικών διαδικασιών. Ένα διάγραμμα δραστηριοτήτων συνήθως περιέχει δραστηριότητες(activities),ενέργειες(actions)και μεταβάσεις(transitions).[15]

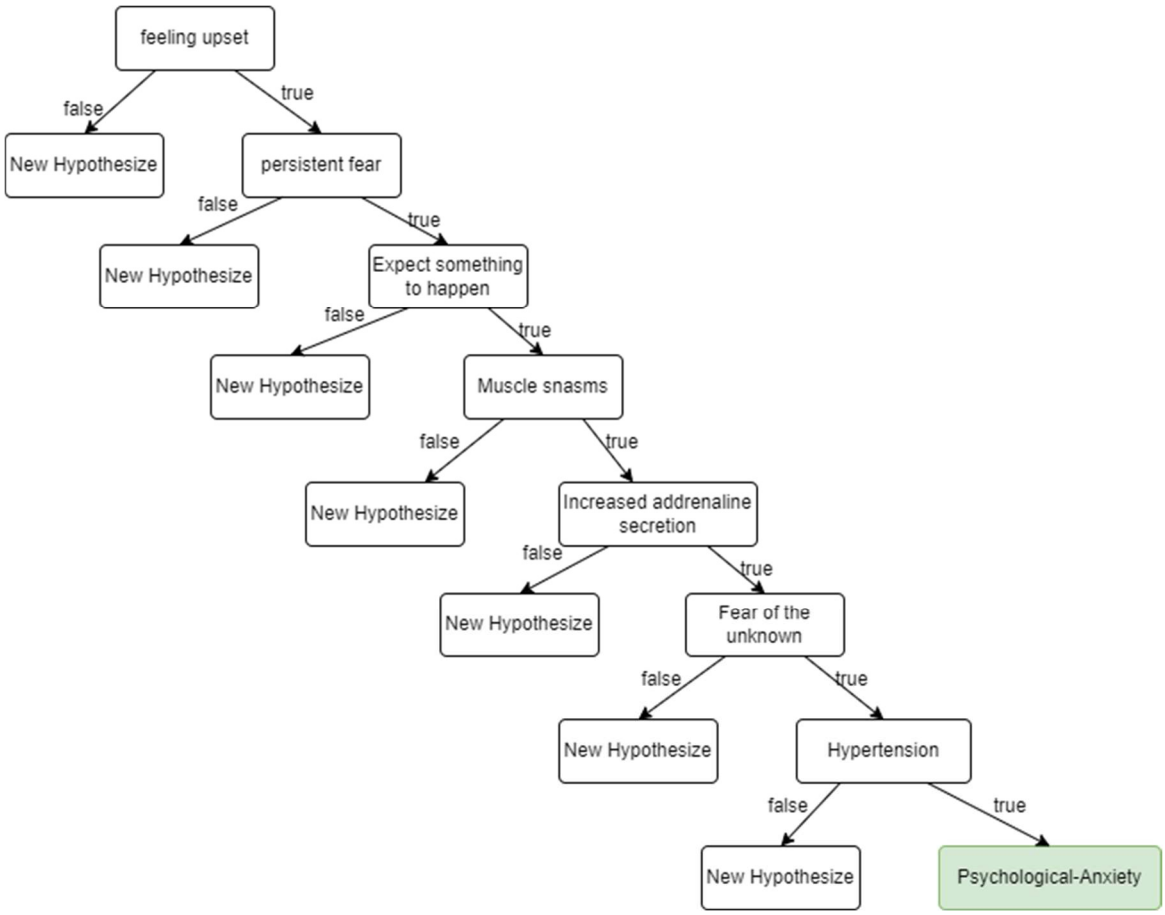
Ενώ το διάγραμμα ER πρόκειται για ένα αφαιρετικό μοντέλο δεδομένων, με καθορισμένη δομή και χρησιμοποιείται για την παροχή εννοιολογικού σχήματος κατά την σχεδίαση βάσεων δεδομένων. Σκοπός του ER είναι να περιγράφει τις αναγκαίες πληροφορίες οι οποίες επρόκειτο να αποθηκευτούν στην βάση δεδομένων. Όταν αναφερόμαστε στον φυσικό σχεδιασμό του μοντέλου σχεδιασμού, πρόκειται για την φάση του λογικού σχεδιασμού όπου το σύστημα δημιουργείται βασισμένο σε μία βάση δεδομένων και το μοντέλο δεδομένων χαρτογραφείται σε προχωρημένο στάδιο μοντελοποίησης. Αφού σχεδιαστεί το λογικό μοντέλο χαρτογραφείται σε κάποιο φυσικό μοντέλο.[16]

2.4 Δέντρα αποφάσεων και Visual Prolog

Το δέντρο αποφάσεων ή αλλιώς υπολογιστική πολυπλοκότητα, πρόκειται για έναν αλγόριθμο οποίος αντιμετωπίζεται ως μια ακολουθία ερωτημάτων ή δοκιμών. Έχει την μορφή δένδρου με ρίζα και κάθε εσωτερικό κόμβο να αντιστοιχεί σε μια ερώτηση. Τα τόξα όπου προέρχονται από τους κόμβους αντιπροσωπεύουν τις πιθανές απαντήσεις, συνήθως είναι της μορφής ναι ή όχι. Ενώ τα φύλλα αντιπροσωπεύουν μια πρόβλεψη της λύσης στο πρόβλημα που εξετάζεται. Το μέγεθος και η ταχύτητα εκτέλεσης αυτού του αλγόριθμου αντιστοιχεί στο βάθος του αντίστοιχου δέντρου αποφάσεων.

Η δημιουργία του δέντρου αποφάσεων χωρίζεται σε δύο κατηγορίες. Στην πρώτη κατηγορία η οποία είναι αυτή της ανάπτυξης, κατασκευάζουμε ένα μεγάλο δένδρο το οποίο απεικονίζει τη βάση δεδομένων με μεγάλη ακρίβεια. Η δεύτερη κατηγορία είναι αυτή του κλαδέματος όπου προσδιορίζεται ακριβώς το μέγεθος του δένδρου, ανάλογα με τους εξειδικευμένους κανόνες οι οποίοι παράγονται πριν την διαδικασία του κλαδέματος.[17][18]

Παρακάτω παρατίθεται το δένδρο αποφάσεων βασισμένο στον κώδικα της εφαρμογής.



Εικόνα 8 Δένδρο Αποφάσεων

Το δένδρο αποφάσεων αυτό βασίζεται στον κώδικα:

```
symptom('Psychological-Anxiety') :-
 positive('Are you', 'feeling upset?'),
 positive('Do you have', 'persistent fear?'),
 positive('Do you', 'except something to happen?'),
 positive('Do you have', 'muscle snasms?'),
 positive('Do you have', 'increased adrenaline secretion?'),
 positive('Do you have', 'fear of the unknown?'),
 positive('Do you have', 'hypertension?').
```

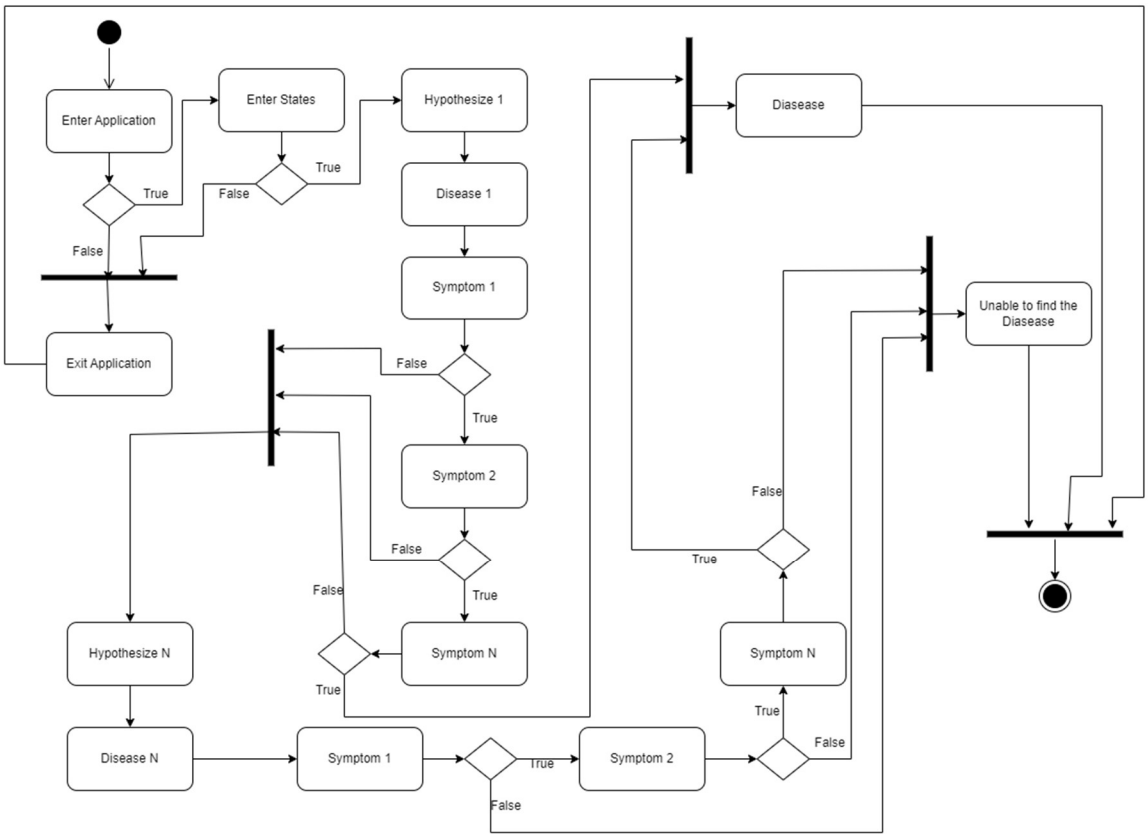
Συγκεκριμένα ο κώδικας αυτός αφορά τις ερωτήσεις όπου κάνει το πρόγραμμα για το ψυχολογικό άγχος. Εάν ο χρήστης απαντήσει θετικά σε όλα τα ερωτήματα τότε η διάγνωση θεωρείται επιτυχής και το αποτέλεσμα είναι Ψυχολογικό Άγχος.

Το κατηγόρημα symptom('Psychological-Anxiety') είναι η υπόθεση του κανόνα, όπου εάν ο χρήστης έχει το σύνολο των συμπτωμάτων τότε πάσχει από ψυχολογικό άγχος.

Ο κανόνας στην ουσία είναι ένα δένδρο αποφάσεων που αποτελείται από ένα σύνολο ερωτήσεων, όπου η καθεμία αντιστοιχεί σε ένα σύμπτωμα που αντιπροσωπεύει το ψυχολογικό άγχος. Οι συνθήκες αυτές αναμένουν δύο ορίσματα, το πρώτο είναι η ερώτηση ναι ή όχι και το δεύτερο είναι η απάντηση του χρήστη.

Αν ο χρήστης απαντήσει θετικά σε όλες τις ερωτήσεις, τότε το κατηγόρημα symptom('Psychological-Anxiety') θεωρείται αληθές και εμφανίζει στον χρήστη πως πάσχει από ψυχολογικό άγχος. Εάν όμως ο χρήστης απαντήσει αρνητικά σε οποιαδήποτε από τις ερωτήσεις, τότε ο κανόνας προχωράει στην επόμενη υπόθεση όπου μπορεί να είναι άλλος κανόνας ή συνθήκη.

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



Εικόνα 9 Διάγραμμα Δραστηριοτήτων

Στο παραπάνω διάγραμμα δραστηριοτήτων αναπαριστάτε η διαδικασία διάγνωσης μιας ασθένειας με τα βήματα που μπορεί να ακολουθήσει ο χρήστης.

Ξεκινάει με τον αρχικό κόμβο και στην συνέχεια εισέρχεται σε κόμβο απόφασης, όπου μπορεί να συνεχίσει στην συμπλήρωση της φόρμας των στοιχείων του ή να προβεί σε έξοδο του προγράμματος. Εάν αποφασίσει να συνεχίσει στην φόρμα συμπλήρωσης και εφόσον συμπληρωθεί, εισέρχεται και πάλι σε κόμβο απόφασης όπου είτε προχωράει στην διαδικασία της διάγνωσης, είτε στην έξοδο του προγράμματος.

Εφόσον αποφασίσει να προβεί στην διαδικασία της διάγνωσης εισέρχεται σε νέο κόμβο απόφασης, όπου ελέγχει αν τα συμπτώματα που του παρουσιάζονται είναι θετικά ή αρνητικά.

Εάν είναι θετικά, το πρόγραμμα προχωράει στην θετική και ελέγχει αν υπάρχει ταύτιση με κάποια από τις γνωστές ασθένειες. Αν υπάρχει ταύτιση, τότε εμφανίζει το όνομα αυτής μέσω ενός πλαίσιο μηνύματος. Ενώ αν δεν υπάρχει ταύτιση, τότε το πρόγραμμα εμφανίζει μήνυμα αδυναμίας εύρεσης ασθένειας.

Στην περίπτωση που τα συμπτώματα είναι αρνητικά, τότε το πρόγραμμα μεταβαίνει σε επόμενη υπόθεση με νέες ερωτήσεις. Η διαδικασία αυτή θα συνεχιστεί έως ότου βρεθούν θετικά

συμπτώματα ώστε να γίνει η διάγνωση, είτε μέχρι να απαντηθούν όλα τα συμπτώματα αρνητικά που σε αυτή την περίπτωση θα εμφανίσει μήνυμα σφάλματος.

Ο τελικός κόμβος ορίζει το τέλος του προγράμματος.

Κεφάλαιο 3: Η Εφαρμογή

Στο τρίτο και τελευταίο κεφάλαιο της εργασίας μας, θα αναλύσουμε την δημιουργία της εφαρμογής τμηματικά τον κώδικα της εφαρμογής. Θα ακολουθήσει η παρουσίαση της με την χρήση στιγμιότυπων, από την πλευρά του χρήστη εξηγώντας την λειτουργικότητα και τα χαρακτηρίστηκα του καταλήγοντας στο αποτέλεσμα.

3.1 Δημιουργία Εφαρμογής

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

Εικόνα 10 ClassFacts

Ξεκινάμε το πρόγραμμα μας δημιουργώντας το classfacts στο οποίο ορίζουμε τις μεταβλητές,xpositive και xnegative. Όπου αποθηκεύουν το σύμπτωμα και το όνομα της ασθένειας αν είναι θετικό στο xpositive ή αν είναι αρνητικό στο xnegative αντίστοιχα.

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

Εικόνα 11 Κανόνας Positive

Συνεχίζουμε με τον κανόνα positive, όπου ελέγχει αν η απάντηση που έδωσε ο χρήστης είναι θετική τότε ο κανόνας είναι επιτυχής. Αν ο χρήστης δώσει αρνητική απάντηση, τότε κάνει στον χρήστη μία ερώτηση σχετικά με το σύμπτωμα.

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

Εικόνα 12 Κανόνας Question

Στον κανόνα questionεμφανίζεται ένα πλαίσιο μηνύματος, όπου κάνει την ερώτηση στον χρήστη σχετικά με κάποιο σύμπτωμα και περιμένει την απάντηση του. Αφού λάβει την απάντηση την αποθηκεύει χρησιμοποιώντας τον κανόνα remember.

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

Εικόνα 13 Κανόνας Remember

Ο κανόνας rememberδέχεται τρείς παραμέτρους την ασθένεια, το σύμπτωμα και την απάντηση του χρήστη. Αν ο χρήστης απαντήσει ‘y’(ναι), τότε αποθηκεύει το σύμπτωμα ως θετικό για την ασθένεια, χρησιμοποιώντας το κατηγόρημα asserta. Ενώ αν απαντήσει ‘n’(όχι), τότε αποθηκεύει ως αρνητικό για την ασθένεια χρησιμοποιώντας ξανά το κατηγόρημα asserta και ο κώδικας αποτυγχάνει. Με τον τρόπο αυτό το σύστημα εξασφαλίζει πως θα εμφανίζει κάθε φορά διαφορετική ερώτηση.

```
clear_facts() :-
   retractFactDb(factDb).
```

Εικόνα 14 Κανόνας ClearFacts

Ο κανόνας clearfacts διαγράφει την βάση δεδομένων χρησιμοποιώντας το κατηγόρημα retractFactDb.

```
clauses
   run() :-
      disease(X),
      !,
      messageBox::displayNote(X, "You may have"),
      clear_facts.

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

Εικόνα 15 Κανόνας Run

Ο κανόνας runκάνει την διάγνωση των ασθενειών με βάση την λίστα συμπτωμάτων. Ελέγχει τις ασθένειες με την σειρά καλώντας τον κανόνα disease. Αν βρει την ασθένεια τότε εμφανίζει ένα μήνυμα με το όνομα της ασθένειας. Αντίθετα αν το σύστημα δεν βρει κάποια

ασθένεια, τότε θα εμφανίσει μήνυμα αποτυχίας εύρεσης ασθένειας. Μετά την εκτέλεση και των δύο περιπτώσεων τα δεδομένα θα διαγραφούν με την χρήση του κανόνα clearfacts.

```
clauses
   disease('Psychological-Anxiety') :-
      symptom('Psychological-Anxiety').

   disease('Obsessive-Compulsive') :-
      symptom('Obsessive-Compulsive').

   disease('Hysteria') :-
      symptom('Hysteria').

   disease('Depression') :-
      symptom('Depression').
```

Εικόνα 16 Κανόνας Diasease

Ο κανόνας diseaseδέχεται μια παράμετρο το όνομα της ασθένειας και στην συνέχεια ελέγχει αν τα συμπτώματα της ασθένειας είναι θετικά με την χρήση του κανόνα symptom.Αν όλα τα συμπτώματα είναι θετικά, τότε θεωρούμε πως έγινε διάγνωση.

```
clauses
   symptom('Psychological-Anxiety') :-
      positive('Are you', 'feeling upset?'),
      positive('Do you have', 'persistent fear?'),
      positive('Do you', 'except something to happen?'),
      positive('Do you have', 'muscle snasms?'),
      positive('Do you have', 'increased adrenaline secretion?'),
      positive('Do you have', 'fear of the unknown?'),
      positive('Do you have', 'hypertension?').

   symptom('Obsessive-Compulsive') :-
      positive('Do you have', 'fear of accumulation of dirt?'),
      positive('Do you have the need to', 'shut the doors continuasly?'),
      positive('Do you have the need to', 'frequently make desissions?'),
      positive('Do you have', 'ideas and questions?'),
      positive('Do you have the need of', 'bathing ten times?'),
      positive('Do you have the need of', 'repeat washing hands?').

   symptom('Hysteria') :-
      positive('Do you have the', 'feeling of hatred?'),
      positive('Do you have', 'absence of consiousness?'),
      positive('Do you have', 'temporary loss of memory?'),
      positive('Do you have', 'misalignment of limbs?').

   symptom('Depression') :-
      positive('Do you have the', 'feeling of hatred?'),
      positive('Do you have', 'food imbalance?'),
      positive('Do you', 'lose hope?'),
      positive('Do you feel', 'inactivity of body?'),
      positive('Do you have', 'life pressures?').
```

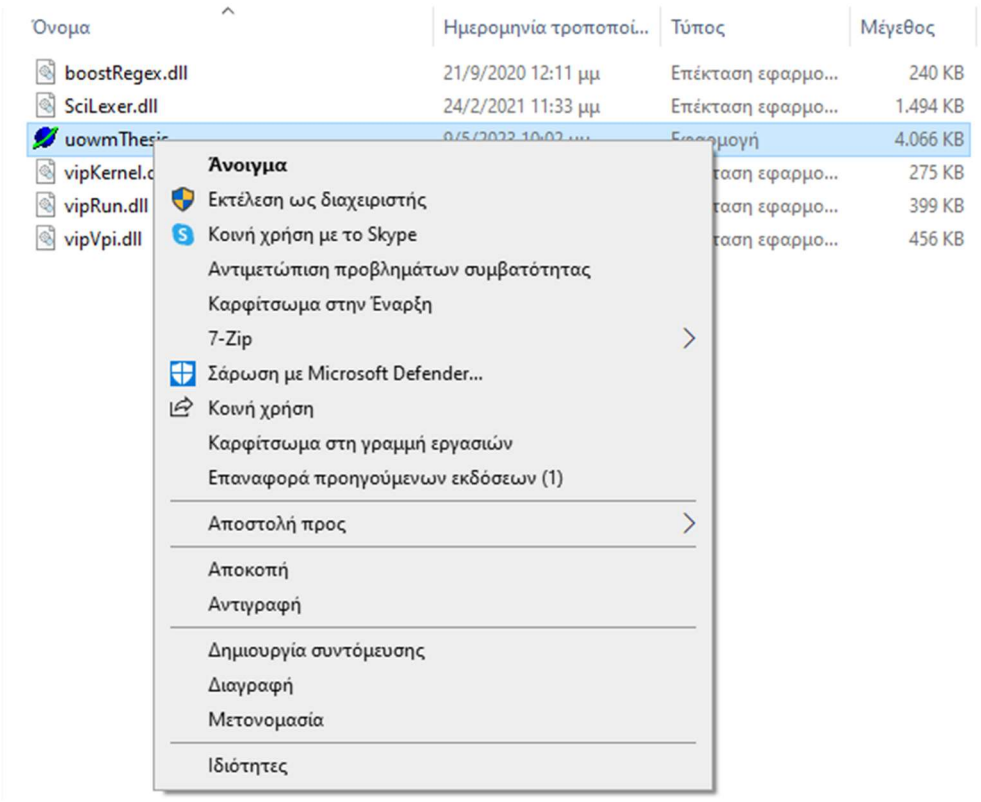
Εικόνα 17 Κανόνας Symptom

Τέλος με τον κανόνα symptomo οποίος δέχεται το όνομα της ασθένειας, ελέγχει τα συμπτώματα αν είναι θετικά με τον κανόνα positive.

3.2 Παρουσίαση Εφαρμογής

Στην παρούσα ενότητα θα γίνει η παρουσίαση της διαγνωστικής μας εφαρμογής, με αναλυτικό βηματισμό που θα ακολουθήσει ο χρήστης στο περιβάλλον της Visual Prolog.

Το πρώτο βήμα που πρέπει να εκτελέσει ο χρήστης είναι, να εκτελέσει το εκτελέσιμο αρχείο που δημιουργήθηκε από το γραφικό περιβάλλον.

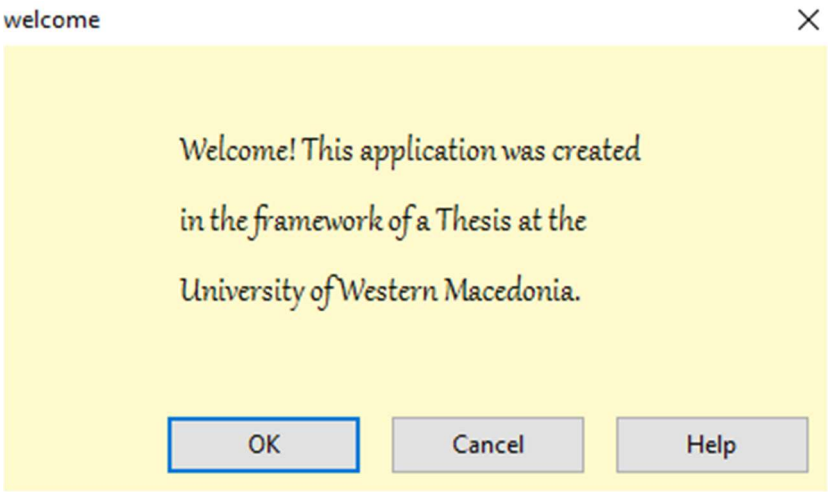


Εικόνα 18 Άνοιγμα της εφαρμογής

Στην συνέχεια θα του εμφανίσει το κύριο παράθυρο, όπου ο χρήστης θα επιλέξει File->New.

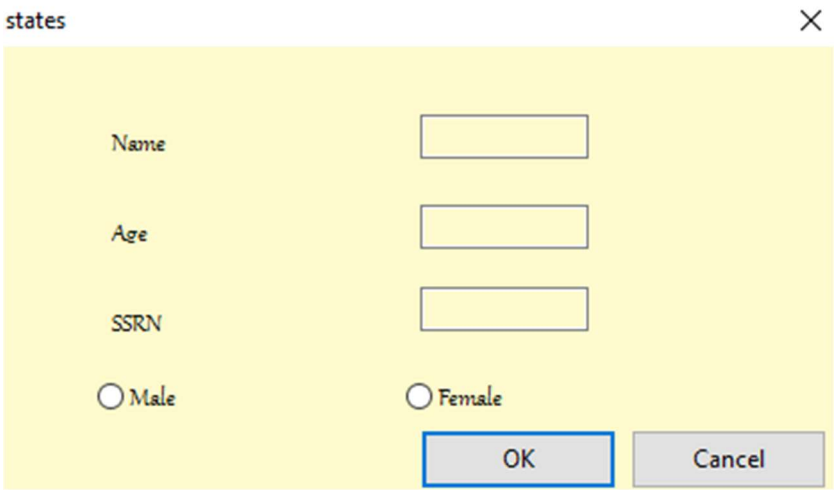
Εικόνα 19 Εκτέλεση της Εφαρμογής

Αφού επιλέξει το Newτότε το σύστημα θα του εμφανίσει το παράθυρο καλωσορίσματος (welcome), όπου μπορεί είτε να συνεχίσει πατώντας το κουμπί OK, είτε να αποχωρήσει πατώντας το κουμπί Cancel.



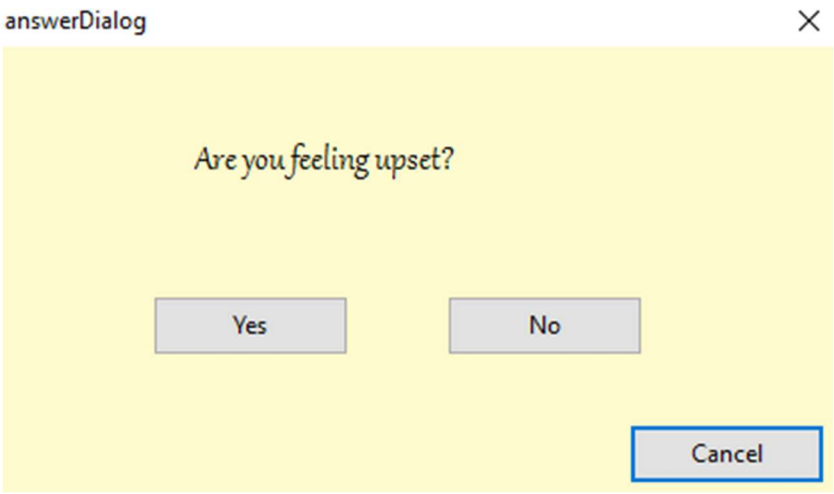
Εικόνα 20 Φόρμα Καλωσορίσματος

Εφόσον επιλέξει το OK του εμφανίζει την φόρμα συμπλήρωσης προσωπικών στοιχείων (states).



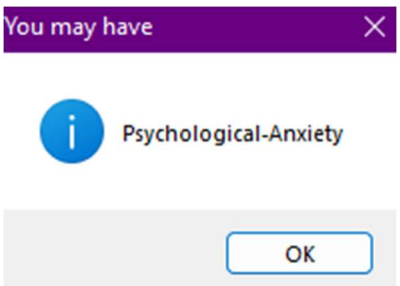
Εικόνα 21 Φόρμα συμπλήρωσης στοιχείων

Αφού συμπληρώσει τα στοιχεία του θα του εμφανίσει την πρώτη ερώτηση από την πρώτη ασθένεια.



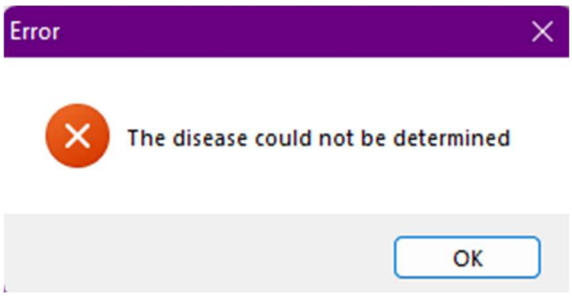
Εικόνα 22 Παράθυρο Διαλόγου

Με βάση τις απαντήσεις του χρήστη θα του εμφανίσει την διάγνωση της ασθένειας ανάλογα με τα συμπτώματα του.



Εικόνα 23 Μήνυμα Διάγνωσης

Υπάρχει όμως και η περίπτωση ο χρήστης να μην πάσχει από κάποια ασθένεια σε αυτή την περίπτωση θα του εμφανίσει το μήνυμα σφάλματος διάγνωσης ασθένειας.



Εικόνα 24 Μήνυμα Σφάλματος

Συμπεράσματα

Η παρούσα πτυχιακή εργασία έχει ως σκοπό της, τον σχεδιασμό και την δημιουργία μιας διαγνωστικής εφαρμογής για ψυχολογικές διαταραχές με την χρήση της γλώσσας Prolog.

Η δημιουργία μιας διαγνωστικής εφαρμογής έχει αρκετές δυσκολίες, καθώς ο προγραμματιστής πρέπει πρώτα να συλλέξει τα δεδομένα που χρειάζεται και στην συνέχεια να τα κατατάσσει σε μια σωστή σειρά, ώστε να τα καταχωρήσει στο πρόγραμμα του με ευκολία.

Για την δημιουργία της εφαρμογής αυτής, χρειάστηκε να μάθω και να κατανοήσω τόσο την γλώσσα της Visual Prolog,όσο και το προγραμματιστικό της περιβάλλον. Τα οποία μελέτησα εις βάθος, ώστε να ολοκληρωθεί η παρούσα πτυχιακή.

Παρόλο που η εφαρμογή ολοκλήρωσε τον σκοπό της και είναι λειτουργική, θα μπορούσε να αναπτυχθεί στο μέλλον. Για αρχή θα μπορούσαν να εισαχθούν παραπάνω ψυχολογικές ασθένειες μαζί με τα συμπτώματα τους, ο χρήστης να δημιουργεί προφίλ για να μπορέσει να χρησιμοποιήσει την εφαρμογή όπου τα στοιχεία του θα ελέγχονται εάν είναι σωστά αλλά να υπάρχει και ειδικό προφίλ που θα μπορεί να δημιουργεί και να χρησιμοποιεί ένας ειδικά εκπαιδευμένος στην ψυχική υγεία.

Βιβλιογραφία

[1] M. McGee, «What is Diagnostic Software?,» 09 /03 /2023. [Ηλεκτρονικό]. Available: https://www.easytechjunkie.com/what-is-diagnostic-software.htm. [Πρόσβαση 10 /05 /2023].

[2] T. O. R. Center, «Diagnostic Testing for Mental Health Disorders,» 10 /04 /2023. [Ηλεκτρονικό]. Available: https://www.orlandorecovery.com/co-occurring-disorders-assessment-treatment/diagnostic-testing/. [Πρόσβαση 10 /05 /2023].

[3] Σ. Καλημέρης, «Ψυχίατρος Σπύρος Καλημέρης,» [Ηλεκτρονικό]. Available: https://kalimeristherapist.com/%CE%A8%CF%85%CF%87%CE%B9%CE%BA%CE%AD%CF%82-%CE%94%CE%B9%CE%B1%CF%84%CE%B1%CF%81%CE%B1%CF%87%CE%AD%CF%82/. [Πρόσβαση 11 /05 /2023].

[4] Wikipedia, «Ψυχική ασθένεια,» /2023.[Ηλεκτρονικό].

Available: https://en.wikipedia.org/wiki/Mental_disorder. [Πρόσβαση 11/05/2023].

[5] Wikipedia, «Programming tool,» /2023. [Ηλεκτρονικό].

Available: https://en.wikipedia.org/wiki/Programming_tool. [Πρόσβαση 19/05/2023].

[6] Wikipedia, «Prolog,» /2023. [Ηλεκτρονικό].

Available: https://en.wikipedia.org/wiki/Prolog.[Πρόσβαση 12/05/2023]

[7] tutorialspoint, «Prolog - Basics,» [Ηλεκτρονικό].

Available: https://www.tutorialspoint.com/prolog/prolog_basics.htm.

[Πρόσβαση 12 /05 /2023].

[8] «MIND DIAGNOSTICS .ORG,» 2022. [Ηλεκτρονικό]. Available: https://www.mind-diagnostics.org/. [Πρόσβαση 20 /05 /2023].

[9] «PSYCHOLOGICAL TESTING ONLINE,» [Ηλεκτρονικό]. Available: https://psytests.org/en.html. [Πρόσβαση 20 /05 /2023].

[10] «MONSENSO,» 2023. [Ηλεκτρονικό]. Available: https://www.monsenso.com/.

[Πρόσβαση 20 /05 /2023].

[11] codegrip, «TOP 10 IDE EVERY DEVELOPER SHOULD KNOW».[Ηλεκτρονικό].

Available:https://www.codegrip.tech/productivity/top-10-ide-every-developer-should-know/.[Πρόσβαση 19/05/2023]

[12] Γεώργιος Πανσεληνάς, Νικόλαος Αγγελιδάκης, Αφροδίτη Μιχαηλίδη, Χαρίλαος Μπλάτσιος, Σταύρος Παπαδάκης, Γεώργιος Παυλίδης, Ελευθέριος Τζαγκαράκης, Αλέξης Τζωρμπατζάκης, «Εφαρμογές Πληροφορικής (Α Λυκείου),» ΔΙΟΦΑΝΤΟΣ, 2014.Κεφάλαιο 6[Ηλεκτρονικό Βιβλίο].

Available:http://ebooks.edu.gr/ebooks/v/html/8547/2714/Pliroforiki_A-Lykeiou_html-empl/index2_6.html. [Πρόσβαση 19/05/2023]

[13] V. Prolog, «Visual Prolog,»/2021. [Ηλεκτρονικό]. Available:https://wiki.visual-prolog.com/index.php?title=Fundamental_Visual_Prolog. [Πρόσβαση 12 /05 /2023].

[14] Wikipedia, «Visual Prolog,» /2022.[Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Visual_Prolog. [Πρόσβαση 12/05/2023].

[15] Δ. Π. Φιτσιλής, «Τα διαγράμματα UML».[Ηλεκτρονικό] Available: http://edu.eap.gr/pli/pli24_old/B-tomos/Parousiaseis/P03.pdf.[Πρόσβαση 22/05/2023]

[16] Wikipedia, «Μοντέλο Οντοτήτων-Συσχετίσεων,» /2019.[Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model. [Πρόσβαση 22/05/2023]

[17] Κ. Σ. Ζήμερας, «ΔΕΝΔΡΑ ΑΠΟΦΑΣΕΩΝ,» 2021.[Ηλεκτρονικό].

Available:https://eclass.aegean.gr/modules/document/file.php/SAS287/ΔΙΑΦΑΝΕΙΕΣ/ΔΕΝΔΡΑ%20ΑΠΟΦΑΣΕΩΝ%20new.pdf

[Πρόσβαση 22/05/2023]

[18] Wikipedia, «Μοντέλο δέντρου απόφασης,» /2022. [Ηλεκτρονικό].Available: https://en.wikipedia.org/wiki/Decision_tree_model [Πρόσβαση 22/05/2023]

Παράρτημα Κώδικα

```
% Copyright

implement main

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

end implement main

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


% Copyright

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) :-
        _ = welcome::display(This).

% This code is maintained automatically, do not update it manually.
predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("uowmThesis"),
        setDecoration(titlebar([frameDecoration::closeButton, frameDecoration::maximizeButton, fr
ameDecoration::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

implement welcome inherits dialog
    open core, vpiDomains

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

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

predicates
    onOkClick : button::clickResponder.
clauses
    onOkClick(_Source) = button::defaultAction :-
```

```
        _ = states::display(getParent()).

% This code is maintained automatically, do not update it manually.
facts
    ok_ctl : button.
    cancel_ctl : button.
    help_ctl : button.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("welcome"),
        setRect(rct(50, 40, 290, 160)),
        setModal(true),
        setDecoration(titlebar([frameDecoration::closeButton])),
        ok_ctl := button::newOk(This),
        ok_ctl:setText("&OK"),
        ok_ctl:setPosition(48, 98),
        ok_ctl:setSize(56, 16),
        ok_ctl:defaultHeight := false,
        ok_ctl:setAnchors([control::right, control::bottom]),
        ok_ctl:setClickResponder(onOkClick),
        cancel_ctl := button::newCancel(This),
        cancel_ctl:setText("Cancel"),
        cancel_ctl:setPosition(112, 98),
        cancel_ctl:setSize(56, 16),
        cancel_ctl:defaultHeight := false,
        cancel_ctl:setAnchors([control::right, control::bottom]),
        help_ctl := button::new(This),
        help_ctl:setText("&Help"),
        help_ctl:setPosition(176, 98),
        help_ctl:setSize(56, 16),
        help_ctl:defaultHeight := false,
        help_ctl:setAnchors([control::right, control::bottom]),
        StaticText_ctl = textControl::new(This),
```

```
        StaticText_ctl:setText("Welcome! This application was created in the framework of a Thesis
 at the University of Western Macedonia."),
        StaticText_ctl:setPosition(52, 18),
        StaticText_ctl:setSize(132, 60),
        StaticText_ctl:setFont(vpi::fontCreateByName("Gabriola", 14)).
% end of automatic code

end implement welcome


implement states inherits dialog
    open core, vpiDomains

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

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

predicates
    onOkClick : button::clickResponder.
clauses
    onOkClick(_Source) = button::defaultAction() :-
        askSymptoms::run().

% This code is maintained automatically, do not update it manually.
facts
    ok_ctl : button.
    cancel_ctl : button.
    name_ctl : editControl.
    age_ctl : editControl.
```

```
    ssrn_ctl : editControl.
    male_ctl : radioButton.
    female_ctl : radioButton.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("states"),
        setRect(rct(50, 40, 290, 160)),
        setModal(true),
        setDecoration(titlebar([frameDecoration::closeButton])),
        ok_ctl := button::newOk(This),
        ok_ctl:setText("&OK"),
        ok_ctl:setPosition(120, 102),
        ok_ctl:setSize(56, 16),
        ok_ctl:defaultHeight := false,
        ok_ctl:setAnchors([control::right, control::bottom]),
        ok_ctl:setClickResponder(onOkClick),
        cancel_ctl := button::newCancel(This),
        cancel_ctl:setText("Cancel"),
        cancel_ctl:setPosition(180, 102),
        cancel_ctl:setSize(56, 16),
        cancel_ctl:defaultHeight := false,
        cancel_ctl:setAnchors([control::right, control::bottom]),
        StaticText_ctl = textControl::new(This),
        StaticText_ctl:setText("Name"),
        StaticText_ctl:setPosition(32, 18),
        StaticText_ctl:setFont(vpi::fontCreateByName("Gabriola", 11)),
        StaticText1_ctl = textControl::new(This),
        StaticText1_ctl:setText("Age"),
        StaticText1_ctl:setPosition(32, 42),
        StaticText1_ctl:setFont(vpi::fontCreateByName("Gabriola", 11)),
        StaticText2_ctl = textControl::new(This),
        StaticText2_ctl:setText("SSRN"),
        StaticText2_ctl:setPosition(32, 66),
```

```
        StaticText2_ctl:setFont(vpi::fontCreateByName("Gabriola", 11)),
        name_ctl := editControl::new(This),
        name_ctl:setPosition(120, 18),
        age_ctl := editControl::new(This),
        age_ctl:setPosition(120, 42),
        ssrn_ctl := editControl::new(This),
        ssrn_ctl:setPosition(120, 64),
        male_ctl := radioButton::new(This),
        male_ctl:setText("Male"),
        male_ctl:setRect(vpiDomains::rct(28, 84, 86, 96)),
        male_ctl:setFont(vpi::fontCreateByName("Gabriola", 11)),
        female_ctl := radioButton::new(This),
        female_ctl:setText("Female"),
        female_ctl:setRect(vpiDomains::rct(116, 84, 174, 96)),
        female_ctl:setFont(vpi::fontCreateByName("Gabriola", 11)).
% end of automatic code

end implement states


% Copyright

implement askSymptoms
    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, " ", Y, "\n")),
        C = string::charLower(string::frontChar(Reply)),
        remember(X, Y, C).

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

    clear_facts() :-
        retractFactDb(factDb).

clauses
    run() :-
        disease(X),
        !,
        messageBox::displayNote(X, "You may have"),
        clear_facts.

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

clauses
    disease('Psychological-Anxiety') :-
        symptom('Psychological-Anxiety').

    disease('Obsessive-Compulsive') :-
```

```
        symptom('Obsessive-Compulsive').

    disease('Hysteria') :-
        symptom('Hysteria').

    disease('Depression') :-
        symptom('Depression').

clauses
    symptom('Psychological-Anxiety') :-
        positive('Are you', 'feeling upset?'),
        positive('Do you have', 'persistent fear?'),
        positive('Do you', 'except something to happen?'),
        positive('Do you have', 'muscle snasms?'),
        positive('Do you have', 'increased adrenaline secretion?'),
        positive('Do you have', 'fear of the unknown?'),
        positive('Do you have', 'hypertension?').

    symptom('Obsessive-Compulsive') :-
        positive('Do you have', 'fear of accumulation of dirt?'),
        positive('Do you have the need to', 'shut the doors continuasly?'),
        positive('Do you have the need to', 'frequently make desissions?'),
        positive('Do you have', 'ideas and questions?'),
        positive('Do you have the need of', 'bathing ten times?'),
        positive('Do you have the need of', 'repeat washing hands?').

    symptom('Hysteria') :-
        positive('Do you have the', 'feeling of hatred?'),
        positive('Do you have', 'absence of consiousness?'),
        positive('Do you have', 'temporary loss of memory?'),
        positive('Do you have', 'misalignment of limbs?').

    symptom('Depression') :-
        positive('Do you have the', 'feeling of hatred?'),
        positive('Do you have', 'food imbalance?'),
        positive('Do you', 'lose hope?'),
```

```
        positive('Do you feel', 'inactivity of body?'),
        positive('Do you have', 'life pressures?').

end implement askSymptoms


% Copyright

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(),
        setBackgroundColor(color_lemonChiffon).

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

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

% This code is maintained automatically, do not update it manually.
facts
    cancel_ctl : button.
    question_ctl : textControl.
    yes_ctl : button.
    no_ctl : button.

predicates
    generatedInitialize : ().
clauses
    generatedInitialize() :-
        setText("answerDialog"),
        setRect(rct(50, 40, 290, 160)),
        setModal(true),
        setDecoration(titlebar([frameDecoration::closeButton])),
        cancel_ctl := button::newCancel(This),
        cancel_ctl:setText("Cancel"),
        cancel_ctl:setPosition(180, 100),
        cancel_ctl:setSize(56, 16),
```

```
        cancel_ctl:defaultHeight := false,
        cancel_ctl:setAnchors([control::right, control::bottom]),
        question_ctl := textControl::new(This),
        question_ctl:setText("Static text"),
        question_ctl:setPosition(56, 20),
        question_ctl:setWidth(104),
        question_ctl:setHeight(34),
        question_ctl:setFont(vpi::fontCreateByName("Gabriola", 14)),
        yes_ctl := button::new(This),
        yes_ctl:setText("Yes"),
        yes_ctl:setPosition(44, 66),
        yes_ctl:setClickResponder(onYesClick),
        no_ctl := button::new(This),
        no_ctl:setText("No"),
        no_ctl:setPosition(128, 66),
        no_ctl:setClickResponder(onNoClick).
% end of automatic code

end implement answerDialog
```