Ατομική Διπλωματική Εργασία

ΠΡΟΤΑΣΗ ΓΙΑ ΔΗΜΙΟΥΡΓΙΑ CHATBOT ΓΙΑ ΥΠΟΣΤΗΡΗΞΗ ΤΟΥ PATIENT SUMMARY ΚΑΙ ΤΟΥ E-PRESCRIPTION

Άλικη Βασίλη

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΤΑΣΗ ΓΙΑ ΔΗΜΙΟΥΡΓΙΑ CHATBOT ΓΙΑ ΥΠΟΣΤΗΡΗΞΗ ΤΟΥ PATIENT SUMMARY ΚΑΙ ΤΟΥ E-PRESCRIPTION

Αλίκη Βασίλη

Επιβλέπων Καθηγητής

Κωνσταντίνος Παττίχης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Σεπτέμβριος 2022 – Μάιος 2023

Ευχαριστίες

Αρχικά, θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή κύριο Κωνσταντίνο Παττίχη, που μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα που σχετίζεται με το πώς η Πληροφορική σε συνεργασία με την Ιατρική μπορούν να προσφέρουν γρήγορη, εύκολη και στοχευμένη υγειονομική περίθαλψη.

Ένα τεράστιο ευχαριστώ οφείλω και στην κ. Ειρήνη Σχίζα, συνεργάτιδα του κ.Παττίχη, η οποία ήταν δίπλα μου καθόλη τη διάρκεια της διπλωματικής εργασίας. Η καθοδηγηση, η βοήθεια και η στήριξη της ήταν μεγάλης σημασίας για τη διεκπεραίωση της διπλωματικής μου εργασίας, αφού οι κατευθυντήριες γραμμές και τα βοηθήματα που μου παρείχε συνέβαλαν ώστε να έχουμε το καλύτερο δυνατό αποτέλεσμα. Ήταν πάντοτε πρόθυμη και διαθέσιμη να μου λύσει οποιεσδήποτε απορίες είχα και για να με βοηθήσει να λύσουμε τα προβλήματα που προέκυπταν κατά τη διάρκεια της υλοποίησης του συστήματος.

Τέλος, θα ήθελα να εκφράσω τις ευχαριστίες μου στην ομάδα του e-Health lab που μου έδειξε εμπιστοσύνη και μου παρείχε τον εξοπλισμό για την πλήρη υλοποίηση της εργασίας. Συγκεκριμένα, θερμές ευχαριστίες στον κ. Ζήνωνα Αντωνίου για την καθοδήγηση και για τον χρόνο που αφιέρωσε για να με βοηθήσει στην κατανόηση της βάσης δεδομένων που χρησιμοποιήθηκε, καθώς και στο να μου εξηγήσει τα ιατρικά έγγραφα. Η βοήθεια του ήταν άμεση, ουσιαστική και χωρίς αυτή η ολοκλήρωση της εργασίας θα ήταν δυσκολότερη. Επίσης, θα ήθελα να ευχαριστήσω και τον κ. Παναγιώτη Σάββα, ο οποίος με βοήθησε σε τεχνικά θέματα και προβλήματα σε σχέση με το πώς να τεθεί σε λειτουργία το chatbot που υλοποιήθηκε και πώς να μπορέσουμε να στήσουμε ένα περιβάλλον δοκιμής για τον έλεγχό του.

Περίληψη

Η παρούσα Ατομική Διπλωματική Εργασία αφορά στη δημιουργία ενός chatbot για ιατρική εκπαίδευση και υγειονομική περίθαλψη. Πιο συγκεκριμένα, το chatbot απευθύνεται σε κάθε άτομο και του δίνει τη δυνατότητα για γρήγορη και εύκολη πρόσβαση στο Patient Summary και το e-Prescription του.

Το Patient Summary και το e-Prescription είναι ιατρικά έγγραφα που περιέχουν προσωπικά ιατρικά δεδομένα, τα οποία αποτελούν μια περίληψη του ιατρικού ιστορικού κάθε ατόμου. Χρησιμοποιούνται κυρίως σε περίπτωση απροσδόκητης ή μη προγραμματισμένης ιατρικής κατάστασης για ενημέρωση των επαγγελματιών υγείας που αναλαμβάνουν την υγειονομική περίθαλψη.

Μέχρι σήμερα, πρόσβαση στα εν λόγω έγγραφα έχουν οι γιατροί και οι φαρμακοποιοί της χώρας μας μέσω του συστήματος υγείας. Είναι απαραίτητο, όμως, να έχουν πρόσβαση σε περίπτωση ανάγκης και γιατροί του εξωτερικού. Για τον λόγο αυτό, αναπτύχθηκε τόσο ιστοσελίδα όσο και εφαρμογή που παρέχει τις πληροφορίες αυτές σε διεθνές επίπεδο, αν αυτό είναι απαραίτητο. Το chatbot που έχει αναπτυχθεί για την παρούσα ατομική διπλωματική εργασία, αποτελεί μια μελλοντική προσθήκη σε ήδη υπάρχοντα συστήματα, όπως είναι το σύστημα Endorse, για ενημέρωση παρόχων υγείας στο εξωτερικό, ώστε ένας ασθενής να μπορεί να απαντήσει σε οποιαδήποτε ερώτηση σχετικά με το ιατρικό ιστορικό του στον εν λόγω πάροχο για την αποφυγή ιατρικών λαθών και τη σωστή υγειονομική του περίθαλψη.

Πιο συγκεκριμένα, μέσω του συστήματος αυτού, υπάρχει δυνατότητα σύνδεσης ενός ασθενή για πρόσβαση στο chatbot. Το chatbot είναι σε θέση να απαντήσει οποιαδήποτε ερώτηση σχετικά με τις πληροφορίες που περιέχει τόσο το Patient Summary όσο και το e-Prescription.

Για την υλοποίηση του chatbot χρησιμοποιήθηκαν η πλατφόρμα BotPress σε συνδυασμό με τη γλώσσα προγραμματισμού JavaScript, αρχεία Jason και βάση δεδομένων mySQL από την οποία αντλούνται τα δεδομένα. Το BotPress είναι ένα open source machine learning framework.

Στο παρόν έγγραφο αναλύεται πλήρως και σε βάθος τόσο η έρευνα που έγινε για την απόκτηση των απαραίτητων γνώσεων σχετικά με το θέμα, όσο και ο τρόπος με τον οποίο χτίστηκε και τέθηκε σε πειραματική λειτουργία το εν λόγω σύστημα. Επίσης, αναλύονται τα αποτελέσματα από την ανατροφοδότηση που έγινε μετά από δοκιμή του συστήματος από διάφορους εθελοντές καθώς και το κατά πόσο θα είναι όντως χρήσιμη και βοηθητική η εν λόγω αναβάθμιση.

Περιεχόμενα

Κεφάλαιο 1 - Εισαγωγή

1.1 Τι είναι το Patient Summary και το e-Prescription

Το Patient Summary και το e-Prescription είναι ιατρικά ψηφιακά έγγραφα τα οποία περιλαμβάνουν βασικά ιατρικά δεδομένα που είναι προσβάσιμα σε διασυνοριακό επίπεδο. Συγκεκριμένα, το Patient Summary είναι ένα τυποποιημένο έγγραφο που περιέχει βασικά κλινικά δεδομένα, όπως τις πρόσφατες εγχειρήσεις και τυχόν αλλεργίες, τα οποία αποτελούν σημαντικά στοιχεία που απαιτούνται για την διασφάλιση ασφαλούς υγειονομικής περίθαλψης. Αυτή η συνοπτική εκδοχή των ιατρικών δεδομένων του ασθενούς δίνει στους επαγγελματίες υγείας τις βασικές πληροφορίες που χρειάζονται για να παρέχουν την απαραίτητη φροντίδα σε διασυνοριακό επίπεδο, τόσο σε απροσδόκητες περιπτώσεις όπως κάποιο ατύχημα όσο και σε περιπτώσεις προγραμματισμένης ιατρικής φροντίδας, όπως μετακίνηση κάποιου ασθενούς στο εξωτερικό για παροχή ιατρικής φροντίδας ή και ως σημείο αποκρυστάλλωσης φακέλων υγείας [20].

Στο e-Prescription καταχωρούνται, επεξεργάζονται και φυλάσσονται ιατρικές συνταγές με την χρήση κάποιου υπολογιστικού συστήματος και ιατρικού λογισμικού, ώστε να έχουν πρόσβαση σε αυτό τόσο οι γιατροί όσο και οι φαρμακοποιοί [11]. Πιο συγκεκριμένα, αποτελεί εγχειρίδιο αναφοράς μεταξύ γιατρών και φαρμακοποιών για βελτίωση της ποιότητας της φροντίδας ασθενών. Περιέχει ακριβείς, και κατανοητές ιατρικές συνταγές για τη σωστή παροχή φαρμακευτικής αγωγής [8].

1.2 Τεχνητή Νοημοσύνη και υγεία

Η Τεχνητή Νοημοσύνη είναι ένας κλάδος της Πληροφορικής ο οποίος τα τελευταία χρόνια βρίσκεται στο προσκήνιο και δίνει λύσεις σε προβλήματα που απασχολούν πολλούς διαφορετικούς τομείς, συμπεριλαμβανομένου και του τομέα της υγεία. Ανέκαθεν η επιστήμη

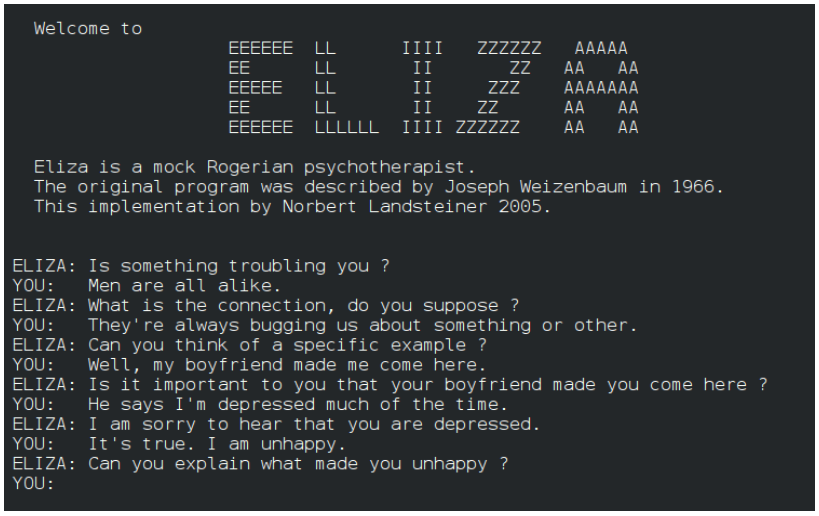
της Πληροφορικής συνέβαλε στην ανάπτυξη του τομέα υγείας με τρανταχτά παραδείγματα τον μαγνητικό τομογράφο, το PET SCAN και πιο πρόσφατα την ανάπτυξη ρομποτικής χειρουργικής. Έτσι, αναπόφευκτα θα εισερχόταν στον τομέα της υγείας και η Τεχνητή Νοημοσύνη με όλες τις μορφές της, αφού η χρήση της μπορεί να διευκολύνει τη διαχείριση δεδομένων υγείας και ιατρικών φακέλων, καθώς και τη διεξαγωγή ερευνών με βάση ιατρικά δεδομένα.

1.3 Chatbots – Ιστορική Αναδρομή

Αυτή η διπλωματική εργασία επικεντρώνεται στο Conversational Artificial Intelligence, το οποίο είναι μια συνθετική δύναμη που δίνει τη δυνατότητα σε μηχανές να καταλαβαίνουν, να επεξεργάζονται και να αντιδρούν στην ανθρώπινη γλώσσα [4]. Πιο συγκεκριμένα, η παρούσα εργασία ασχολείται με chatbots, τα οποία αποτελούν μια μορφή conversational AI.

Η ιστορία των bots, ξεκινά το 1950, όταν ο Βρετανός επιστήμονας Alan Turing, έθεσε το ερώτημα «Μπορούν οι μηχανές να σκεφτούν;» και δημοσίευσε το άρθρο «Υπολογιστική Μηχανή και Νοημοσύνη» [21]. Ακολούθησε το γνωστό σε όλους Turing Test, το οποίο αποτελεί μια δοκιμή της ικανότητας μιας μηχανής να επιδεικνύει ευφυή συμπεριφορά ισοδύναμη με αυτή του ανθρώπου [10]. Όπως αναφέρει ο Alan Turing, μια μηχανή είναι έξυπνη αν μπορεί να υποδυθεί τη συμπεριφορά ενός ανθρώπου πείθοντας ένα άλλο άτομο πως συνομιλεί πραγματικά με άνθρωπο [21].

Το έργο του Turing απασχόλησε και ενέπνευσε τον Joseph Weizenbaum, Γερμανό επιστήμονα υπολογιστών, ο οποίος δημιούργησε το πρώτο chatbot, με το όνομα ELIZA, το 1966. Το chatbot αυτό δημιουργήθηκε αρχικά για να δείξει την επιφανειακή επικοινωνία ανθρώπου υπολογιστή την εποχή εκείνη, όμως μετά την τοποθέτηση του σε προσωπικούς υπολογιστές, προσέλκυσε ιδιαίτερα το ενδιαφέρον, αφού οι χρήστες το βρήκα συναρπαστικό και έτσι δημιουργήθηκαν πολλές παραλλαγές του.
Η πιο γνωστή από αυτές ήταν μια παραλλαγή που λειτουργούσε σαν Ψυχοθεραπευτής ο οποίος αφού λάμβανε μια ερώτηση, «επέστρεφε» την ερώτηση στον ασθενή, αναδιατυπώνοντάς την. Η λειτουργία του εν λόγω chatbot δεν βασίζεται στην αυτόματη μάθηση ή στην πραγματική τεχνητή νοημοσύνη, αφού αυτό που κάνει είναι να καταλαβαίνει λέξεις κλειδιά και βάση αυτών και προηγούμενων λέξεων που αναγνώρισε, να βρίσκει από ένα σύνολο απαντήσεων ποια είναι η πιο κατάλληλη επόμενη απάντηση [35].



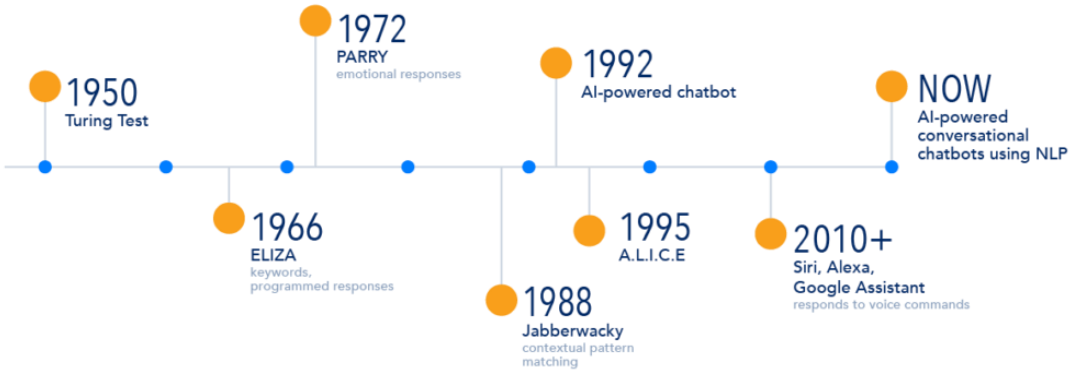
Σχήμα 1.1: παράδειγμα εκτέλεσης του ELIZA

Η ELIZA αποτελεί το πρώτο στην ιστορία chatbot στο οποίο έγινε μια δοκιμή Turing. Όμως, όπως και όλα τα επόμενα προγράμματα, μέχρι και σήμερα, απέτυχε στο να περάσει το Turing Test. Βάση των σημερινών δεδομένων, η ELIZA αποτελεί ένα όχι πολύ έξυπνο σύστημα, αφού αποτυγχάνει πολύ γρήγορα, ήταν όμως ένα αρκετά πρωτοποριακό σύστημα για την εποχή που αναπτύχθηκε.

Μετά το ELIZA ακολούθησαν, πάρα πολλά άλλα. Το 1972, δημιουργήθηκε το chatbot PARRY, από τον ψυχίατρο Kenneth Colby , το οποίο είχε σκοπό να μιμηθεί ένα άτομο που πάσχει από παρανοϊκή σχιζοφρένεια. Τέλη της δεκαετίας του ’80 , ο Βρετανός επιστήμονας Rollo Carpenter δημιούργησε το Jabberwacky, το οποίο χρησιμοποιούσε τεχνικές της Τεχνητής Νοημοσύνης. Το 1995 εμφανίστηκε το A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) το οποίο προσομοιώνει μια γυναίκα που είναι σε θέση να απαντήσει ερωτήσεις που αφορούν τον χαρακτήρα και τα ενδιαφέροντα της.

Προχωρώντας στον χρόνο, και συγκεκριμένα μετά το 2010, το Conversational AI και τα chatbots γίνονται ευρέως γνωστά με την εμφάνιση της Alexa,της Siri, του Google Assistant και πάρα πολλών άλλων, τα οποία πλέον αναγνωρίζουν και φωνή εκτός του γραπτού λόγου.

Σήμερα, σχεδόν όλα τα chatbots χρησιμοποιούν τεχνικές της Τεχνητής Νοημοσύνης σε συνδυασμό με τη Μηχανική Μάθηση, οι οποίες επεξηγούνται και αναλύονται σε επόμενα κεφάλαια της παρούσας εργασίας.



Σχήμα 1.2 : Χρονολογική αναδρομή στην ιστορία των chatbots

1.4 Στόχος Διπλωματικής Εργασίας

Στόχος της Διπλωματικής μου Εργασίας είναι να εξεταστεί κατά πόσο η δημιουργία ενός chatbot θα διευκολύνει τους παρόχους υγείας και τους ασθενείς στη διαδικασία σωστής πληροφόρησης και ενημέρωσης σχετικά με το Patient Summary και το e-Prescription.

Πιο αναλυτικά, στόχος του συστήματος που αναπτύσσεται είναι η δοκιμή της τεχνολογίας αυτής και η διαπίστωση αν όντως αυτό θα αποτελούσε μια έξυπνη και πρακτική προσθήκη σε ήδη υλοποιημένα συστήματα όπως είναι το Endorse (Endorse: https://endorse.cs.ucy.ac.cy/) για ενημέρωση παρόχων υγείας διασυνοριακά, ώστε το ιατρικό ιστορικό κάθε ασθενή να είναι εύκολα και άμεσα προσβάσιμο για να μην χάνεται πολύτιμος χρόνος.

Το chatbot που αναπτύσσεται θα αποτελεί μέσο στο οποίο θα μπορεί να ανατρέξουν τόσο γιατροί όσο και ασθενείς για να αντλήσουν εύκολα και γρήγορα τις απαραίτητες πληροφορίες που χρειάζονται ώστε να υπάρξει μια σωστή και άμεση υγειονομική περίθαλψη.

1.5 Δομή Διπλωματικής Εργασίας

Στο πρώτο κεφάλαιο, γίνεται μια γενική εισαγωγή του θέματος της παρούσας Διπλωματικής εργασίας. Περεταίρω πληροφορίες και εμβάθυνση στις έννοιες που αναφέρονται γίνεται σε επόμενα κεφάλαια. Επίσης, στο κεφάλαιο αυτό παραθέτονται ο σκοπός της Διπλωματικής Εργασίας και η δομή του δοκιμίου.

Στο δεύτερο κεφάλαιο γίνεται περιγραφή του προβλήματος, ορίζεται η ανάγκη που οδήγησε στην απόφαση για την υλοποίηση του εν λόγω συστήματος και αναλύονται επιγραμματικά άλλα παρόμοια συστήματα που υπάρχουν.

Στο τρίτο κεφάλαιο αναλύονται σε βάθος βασικές έννοιες και απαιτούμενες γνώσεις για την υλοποίηση του chatbot. Στο κεφάλαιο αυτό γίνεται, μεταξύ άλλων, αναφορά στη Μηχανική Μάθηση, στην τεχνική NLP και σε βασικές γλώσσες προγραμματισμού, οι οποίες χρησιμοποιήθηκαν στην υλοποίηση του εν λόγω συστήματος.

Στο τέταρτο κεφάλαιο παρουσιάζονται οι απαιτήσεις του συστήματος, τόσο λειτουργικές όσο και μη λειτουργικές, και οι προδιαγραφές του. Αναλύονται σε βάθος οι

αρχές για τη σωστή σχεδίαση ενός chatbot, παραθέτονται οι τύποι χρηστών και τα χαρακτηριστικά τους και επεξηγούνται όλες οι πιθανές αλληλεπιδράσεις που μπορεί να έχει ένας χρήστης με το σύστημα. Επίσης, παρουσιάζεται το Use Case Diagram του συστήματος.

Στο πέμπτο κεφάλαιο επεξηγείται η πλατφόρμα ανάπτυξης, καθώς επίσης και ο τρόπος με τον οποίο σχεδιάστηκε και τέθηκε σε λειτουργία το παρόν σύστημα. Στο κεφαλαίο αυτό δίνονται οι ροές συζήτησης, επεξηγούνται τα βασικά στοιχεία του BotPress και παρουσιάζεται η παρούσα υλοποίηση.

Στο έκτο κεφάλαιο παρουσιάζεται και αναλύεται η φάση δοκιμής και ανατροφοδότησης. Δίνονται η σελίδα δοκιμής, το ερωτηματολόγιο, καθώς επίσης και τα αποτελέσματα που λήφθηκαν. Επίσης, παρουσιάζονται τα συμπεράσματα βάση των αποτελεσμάτων των φάσεων αξιολόγησης.

Τέλος στο έβδομο κεφάλαιο παρουσιάζεται η μελλοντική δουλεία. Αναλύονται πράγματα που θα πρέπει να βελτιωθούν και να ολοκληρωθούν και παρατίθενται ιδέες για μελλοντική αναβάθμιση του παρόντος συστήματος, λαμβάνοντας υπόψη και την ανατροφοδότηση.

Κεφάλαιο 2 – Περιγραφή Προβλήματος/Άλλα συστήματα

2.1 Αναλυτική Περιγραφή Προβλήματος – Ανάγκη

Μέχρι σήμερα, πρόσβαση στο Patient Summary και στο e-Prescription έχουν οι γιατροί και οι φαρμακοποιοί της χώρας μας μέσω του Γενικού Συστήματος Υγείας (ΓΕΣΥ). Με βάση τον κανονισμό της Ευρωπαϊκής Ένωσης είναι απαραίτητο να μπορούν να έχουν πρόσβαση στα έγγραφα αυτά και γιατροί του εξωτερικού σε περιπτώσεις ιατρικής ανάγκης. Ως εκ τούτου, αναπτύχθηκε ήδη τόσο ιστοσελίδα όσο και εφαρμογή που έχει τη δυνατότητα να παρέχει τις πιο πάνω πληροφορίες σε διεθνές επίπεδο.

Το σύστημα που αναπτύσσεται στην παρούσα ατομική διπλωματική εργασία αποτελεί μελλοντική προσθήκη σε ήδη υπάρχοντα συστήματα που έχουν δημιουργηθεί για ενημέρωση παρόχων υγείας στο εξωτερικό, ώστε ένας ασθενής να μπορεί να απαντήσει σε οποιαδήποτε ερώτηση σχετικά με το ιατρικό ιστορικό του για την αποφυγή ιατρικών λαθών και τη σωστή υγειονομική του περίθαλψη.

Παρόλο που υπάρχει η δυνατότητα πρόσβασης ενός ασθενή στο Patient Summary και το e-Prescription μέσω ήδη υλοποιημένων ιστοσελίδων και εφαρμογών, τα έγγραφα αυτά είναι μεγάλα και αρκετά δύσκολα στην ανάγνωση και κατανόηση. Για να βρει κάποιος την πληροφορία που ψάχνει χρειάζεται αρκετός χρόνος. Ως εκ τούτου, δημιουργείται η ανάγκη να μπορεί ένας ασθενής να έχει άμεση και γρήγορη πρόσβαση στα δεδομένα αυτά. Από την εν λόγω ανάγκη προέκυψε και η ιδέα για την ανάπτυξη του chatbot.

Μέσω του chatbot ο ασθενής έχει τη δυνατότητα να θέσει οποιαδήποτε ερώτηση σχετικά με την πληροφορία που ψάχνει και το chatbot να του δώσει κατευθείαν την απάντηση. Έτσι, ο χρόνος που απαιτείται για την εύρεση μιας πληροφορίας από το Patient Summary ή το e-Prescription ελαχιστοποιείται.

2.2 Περιγραφή σεναρίων συζήτησης

Όπως αναφέρθηκε ήδη, στόχος του chatbot είναι η εύκολη και γρήγορη πρόσβαση σε πληροφορίες που περιέχονται στο Patient Summary και το e-Prescription.

Αρχικά, με την είσοδο του χρήστη, το chatbot θα εμφανίζει μήνυμα καλωσορίσματος και ρωτά κατά πόσο ο χρήστης χρειάζεται κάποια βοήθεια.

Αν ο χρήστης απαντήσει θετικά, τότε το chatbot ζητά από τον χρήστη την ταυτότητά του ώστε να μπορέσει να βρει το αντίστοιχο ιατρικό ιστορικό για να είναι σε θέση να απαντήσει τις ερωτήσεις του ασθενή. Αφού ο χρήστης δώσει ορθά την ταυτότητά του, εισέρχεται στην κύρια ροή συζήτησης μέσω της οποίας θα μπορεί να ενημερωθεί για τις πληροφορίες που περιέχονται στο προσωπικό του Patient Summary και e-Prescription.

Όταν ο χρήστης βρίσκεται στο κύριο μονοπάτι συζήτησης μπορεί να απευθύνει στο chatbot οποιαδήποτε ερώτηση θέλει. Αν το chatbot δεν καταλάβει την ερώτηση ή η ερώτηση που υποβάλλει ο χρήστης δεν σχετίζεται με τις πληροφορίες που μπορεί να παρέχει το chatbot, τότε εμφανίζει βοηθητικό μήνυμα για να καθοδηγήσει τον χρήστη να υποβάλει μια έγκυρη ερώτηση.

```
-Hi
-Welcome to our services! Do you need any help?
-Yes
-Please type your id (ex "10101010")
-10101122
-Hello Maria! Ask me your question or type "help" so that i can help you find out what you want.
```

Σχήμα 2.1 : Παράδειγμα εισόδου χρήστη στην κύρια ροή συζήτησης

```
-Do I have any food allergies?
- You have:
  1) Allergy in banana (Reaction: asthma)
  2) Allergy in tomato (Reaction: Cardiac arrhythmia)
```

Σχήμα 2.2 : Παράδειγμα συζήτησης μετά την είσοδο του χρήστη

```
-How is the weather today in Nicosia?
-I am sorry but i do not understand your question.
 I can answer only questions related to your personal patient summary or e-Prescription.
 Ask me your question or type "help" so that i can help you find out what you want.
```

Σχήμα 2.3 : Παράδειγμα άσχετης ερώτησης

2.3 Άλλα Παρόμοια Συστήματα

Με την ανάπτυξη της τεχνολογίας και την εισαγωγή της στον τομέα της υγείας, έχουν αναπτυχθεί πολλές πλατφόρμες με σκοπό την παροχή ιατρικής βοήθειας. Όλες αυτές οι πλατφόρμες αποσκοπούν είτε σε απλή ενημέρωση των ατόμων σχετικά με κάποιο ιατρικό ζήτημα είτε στην παροχή βοήθειας σε κάποιο πάροχο υγείας.

Η Τεχνητή Νοημοσύνη αποτελεί ένα αρκετά ισχυρό εργαλείο για την ανάπτυξη τέτοιων πλατφορμών, αφού παρέχει τη δυνατότητα δημιουργίας «έξυπνων» πρακτόρων, οι οποίοι είναι σε θέση να ανταποκρίνονται καλύτερα και να κατανοούν περισσότερο το

περιβάλλον τους. Παραδείγματα τέτοιων «έξυπνων» πρακτόρων αποτελούν τα chatbots, τα hybrid chats και οι πλατφόρμες προσομοίωσης.

Μέσα από την έρευνα που πραγματοποίησα σχετικά με υπάρχοντα συστήματα στον χώρο της υγείας δεν έχω εντοπίσει κάποιο άλλο σύστημα το οποίο να αναπτύχθηκε με σκοπό όσα αναφέρθηκαν στην εισαγωγή αυτού του δοκιμίου. Στο παρόν στάδιο, υπάρχουν συστήματα τα οποία παρέχουν τις πληροφορίες αυτές, όπως είναι και το ΓΕΣΥ (Γενικό Σύστημα Υγείας) στην Κύπρο, όμως κανένα τους δεν χρησιμοποιεί τεχνικές βασισμένες στην Τεχνητή Νοημοσύνη.

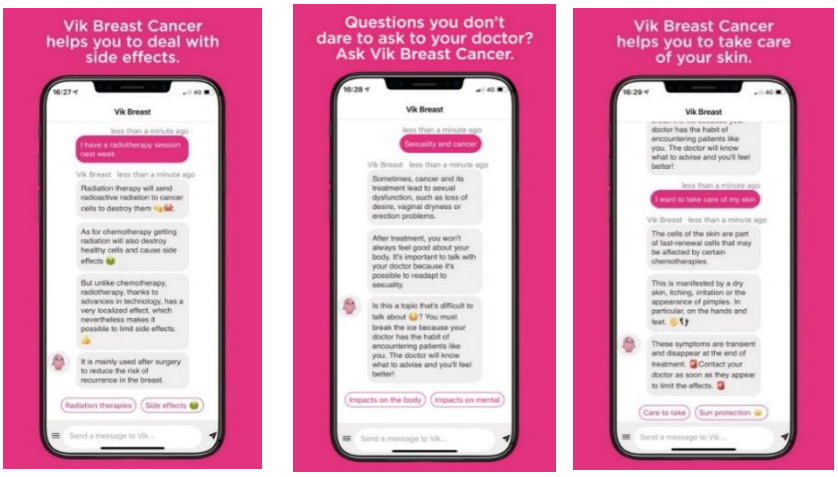
Στη συνέχεια του κεφαλαίου επεξηγούνται επιγραμματικά κάποια υφιστάμενα συστήματα που έχουν σχέση με τον τομέα της υγείας και χρησιμοποιούν τεχνικές της Τεχνητής Νοημοσύνης. Τα συστήματα αυτά αποσκοπούν στην βελτίωση της υγειονομικής περίθαλψης και στην παροχή ιατρικής βοήθειας.

2.3.1 Vik Breast

Το «Vik Breast» είναι ένα σύστημα το οποίο αποτελεί ένα εικονικό «σύντροφο» που βοηθά τον ασθενή άλλα και τους γύρω του σχετικά με την αντιμετώπιση του καρκίνου του μαστού.

Στοχεύει στην καθημερινή παρακολούθηση του ασθενή και στην παροχή συμβουλών από επαγγελματίες υγείας, ιδιαίτερα για τον τρόπο ζωής. Επίσης, διαθέτει τη δυνατότητα ενημέρωσης σχετικά με τον καρκίνο του μαστού, τις θεραπείες που υπάρχουν καθώς ακόμη υπενθυμίζει στον ασθενή τη φαρμακευτική του αγωγή [1].

Το σύστημα αυτό αναπτύχθηκε από τη Wefight, το 2018 και είναι διαθέσιμο έκτοτε σε κάθε άτομο μέσω App Store.



Σχήμα 2.3.1 : Παράδειγμα από συνομιλία στο Vik Breast

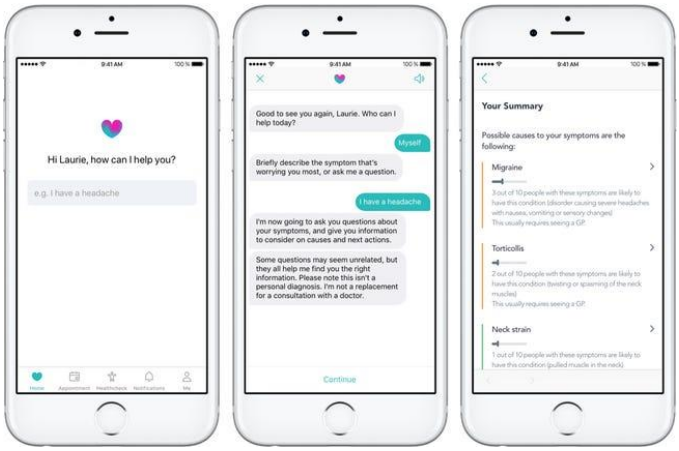
2.3.2 Babylon Health

Το «Babylon Health» προσφέρει εύκολη και γρήγορη πρόσβαση σε γενικούς γιατρούς, φυσιοθεραπευτές, νοσοκόμους και άλλους επαγγελματίες υγείας. Γενικός

στόχος του είναι να φέρει τους ασθενείς σε επαφή με παρόχους υγείας για τη βελτίωση της υγειονομικής τους περίθαλψης.

Ένας ασθενής μπορεί να στείλει ερώτηση ή και φωτογραφία στην ιατρική ομάδα του συστήματος και να έρθει σε άμεση επαφή με κάποιον πάροχο υγείας. Επιπρόσθετα, οι πάροχοι υγείας μπορούν να χορηγήσουν συνταγές για φαρμακευτική αγωγή στον ασθενή ή να τις αποστέλλουν κατευθείαν σε κάποιο φαρμακείο.

Σε περιπτώσεις που η φυσική εξέταση είναι απαραίτητη, τότε ο ασθενής μπορεί να κλείσει ραντεβού μέσω του συστήματος για ιατρική εξέταση στον χώρο της εταιρείας [2].

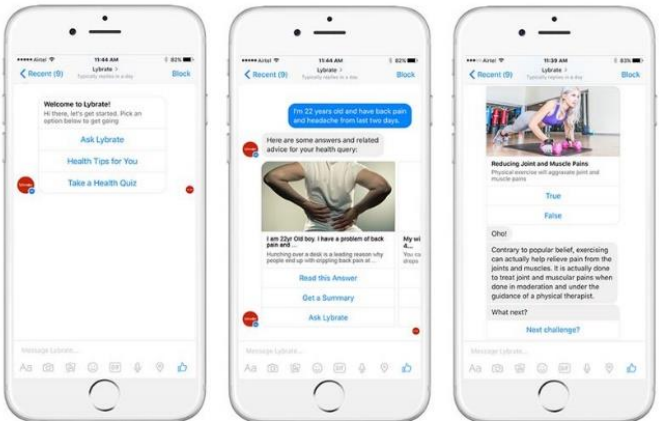


Σχήμα 2.3.2 : Παράδειγμα από συνομιλία στο Babylon Health

2.3.3 Lybrate

Το «Lybrate» είναι εταιρεία υγειονομικής περίθαλψης, η οποία έθεσε σε λειτουργία Facebook Messenger chatbot που είναι σε θέση να απαντήσει σε απλές ερωτήσεις σχετικά με την υγεία του ασθενή.

Ο χρήστης μπορεί να καταγράψει τα προσωπικά του στοιχεία, καθώς και τα συμπτώματα που έχει και στη συνέχεια ένας επαγγελματίας υγείας να του δώσει την απαραίτητη πληροφόρηση. Επίσης, μπορεί αν επιθυμεί, να ζητήσει από το chatbot να τον συνδέσει μέσω τηλεδιάσκεψης με έναν γιατρό από την εταιρεία [26].



Σχήμα 2.3.3 : Παράδειγμα από συνομιλία με το Lybrate chatbot

Κεφάλαιο 3 – Απαιτούμενη Γνώση και Τεχνολογίες

3.1 Βασικοί Ορισμοί

Για την πλήρη κατανόηση του πώς δημιουργείται και λειτουργεί ένα chatbot χρειάζεται να αναλυθούν βασικοί ορισμοί και τεχνολογίες. Σε αυτό το υποκεφάλαιο του δοκιμίου θα αναλυθούν πλήρως αυτοί οι ορισμοί ώστε να γίνει κατανοητή η λογική πίσω από την υλοποίηση και την ανάπτυξη τέτοιων εφαρμογών.

3.1.1 Τεχνητή Νοημοσύνη

Το 2004 ο John McCarthy όρισε την Τεχνητή Νοημοσύνη, ή αλλιώς γνωστή με τον αγγλικό όρο Artificial Intelligence (AI), ως την «επιστήμη και μηχανική για τη δημιουργία ευφυών προγραμμάτων που σχετίζονται με την κατανόηση της ανθρώπινης νοημοσύνης, χωρίς όμως να περιορίζεται σε μεθόδους που να είναι βιολογικά παρατηρήσιμες» [16].

Γενικά, η Τεχνητή Νοημοσύνη είναι μια αρκετά ευρεία έννοια η οποία συνδυάζει πολλούς ορισμούς και τεχνικές και έχει οριστεί με πολλούς διαφορετικούς τρόπους. Όμως, ο όρος αυτός παραπέμπει σχεδόν πάντα στην ικανότητα μιας μηχανής να αναπαράγει τις γνωστικές λειτουργίες ενός ανθρώπου.

Τα συστήματα τεχνητής νοημοσύνης είναι ικανά να «κατανοούν» το περιβάλλον τους και να επιλύουν προβλήματα με σκοπό την επίτευξη ενός στόχου. Επίσης, είναι ικανά, ως έναν βαθμό, να προσαρμόζουν τις αποφάσεις και συμπεριφορές τους βάσει των συνεπειών που είχαν προηγούμενες δράσεις τους, με αποτέλεσμα να λύνουν με αυτονομία πολλά προβλήματα [14].

3.1.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση σύμφωνα με τον Άρθουρ Σάμουελ είναι «πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί» [33].

Γενικά, το πεδίο της Μηχανικής Μάθησης, το οποίο αποτελεί πλέον κομμάτι της Τεχνητής Νοημοσύνης, διερευνά τη μελέτη και κατασκευή αλγορίθμων που είναι ικανοί να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις για αυτά. Όλοι αυτοί οι αλγόριθμοι έχουν σαν κύριο χαρακτηριστικό τη δημιουργία μοντέλων βάσει πειραματικών δεδομένων ώστε να είναι σε θέση να εξάγουν αποφάσεις που εκφράζονται ως αποτελέσματα [23].

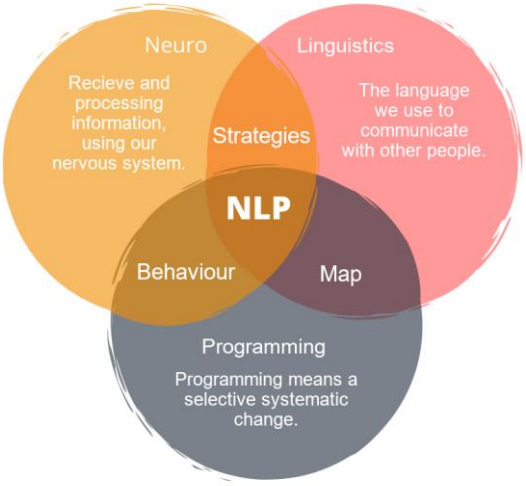
Για τη δημιουργία ενός chatbot, χρησιμοποιούνται τέτοιοι αλγόριθμοι ώστε να επεξεργάζεται σωστά η είσοδος που δίνει ο χρήστης (γραπτός λόγος, φωνητικό μήνυμα κ.τ.λ.) και το bot να «κατανοεί» λέξεις κλειδιά οι οποίες το βοηθούν στο να επιλέξει ποια είναι η καταλληλότερη απόφαση [3].

3.1.3 Natural Language Processing (NLP)

Το NLP (Natural Language Processing – «Επεξεργασία Φυσική Γλώσσας») είναι ένας κλάδος της Πληροφορικής και πιο συγκεκριμένα της Τεχνητής Νοημοσύνης, ο οποίος ασχολείται με την ικανότητα των μηχανών να κατανοήσουν πλήρως τη φυσική γλώσσα. Στοχεύει στο να δώσει στους υπολογιστές την ικανότητα να κατανοούν γραπτά κείμενα και προφορικό λόγο σχεδόν με τον ίδιο τρόπο με τον οποίο μπορούν οι άνθρωποι.

Για να επιτευχθεί κάτι τέτοιο χρησιμοποιούνται αρκετές τεχνικές. Το NLP συνδυάζει την υπολογιστική γλωσσολογία με μοντέλα στατιστικής, μηχανικής και βαθιάς μάθησης. Αυτές οι τεχνολογίες επιτρέπουν στους υπολογιστές να επεξεργάζονται την ανθρώπινη γλώσσα με τη μορφή κειμένου ή φωνητικών δεδομένων και να «κατανοούν» το πλήρες νόημά της, με την πρόθεση και το συναίσθημα του ομιλητή ή του συγγραφέα [17][34].

Το NLP είναι επίσης γνωστό ως Neurolinguistics Programming, ονομασία που προέρχεται από τις τρεις βασικές τεχνικές που χρησιμοποιεί, δηλαδή Neuron (Νευρώνας), Linguistics (Γλωσσολογία) και Programming (Προγραμματισμός) [25].

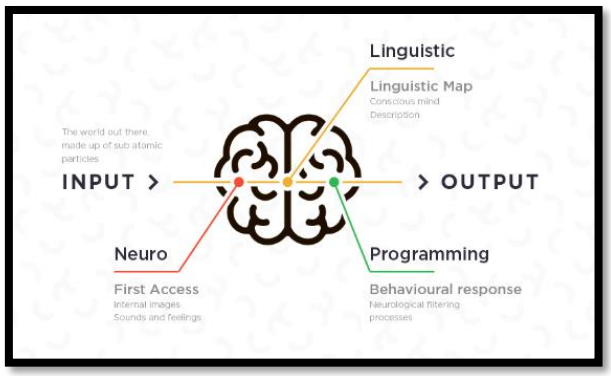


Σχήμα 3.1: NLP Processing

Ο Νευρώνας βασίζεται στο νευρικό μας σύστημα. Η αντίληψη, η σκέψη, τα συναισθήματα και η συμπεριφορά μας εξαρτόνται από το νευρικό μας σύστημα το οποίο επεξεργάζεται δεδομένα από το περιβάλλον μας και διαχειρίζεται τις αντιδράσεις μας σε αυτό. Έτσι και ο Νευρώνας, στην περίπτωση αυτή, στόχο έχει τη συλλογή και επεξεργασία δεδομένων από το περιβάλλον.

Η Γλωσσολογία αντιπροσωπεύει τη γλώσσα με την οποία μπορούν να επικοινωνήσουν οι άνθρωποι, τόσο εξωτερικά, δηλαδή μεταξύ τους, όσο και εσωτερικά, δηλαδή με τον εαυτό τους. Οι λέξεις, η γραμματική, τα μέρη του λόγου, καθώς επίσης και ο τόνος της φωνής συνθέτουν διαφορετικές μεταξύ τους σωστές προτάσεις, οι οποίες μπορούν να έχουν ίδιο ή διαφορετικό νόημα.

Τέλος, ο Προγραμματισμός είναι ουσιαστικά η συμπεριφορική απόκριση, η οποία θα συμβεί στο τέλος της διαδικασίας λόγω των αποτελεσμάτων των προηγούμενων δύο τεχνικών, δηλαδή του νευρολογικού φιλτραρίσματος και του γλωσσικού χάρτη.



Σχήμα 3.2 : NLP procedure

Γενικότερα, η ανθρώπινη γλώσσα είναι γεμάτη με ασάφειες που καθιστούν πολύ δύσκολη τη σύνταξη λογισμικού που να προσδιορίζει με ακρίβεια την επιδιωκόμενη έννοια του κειμένου ή των φωνητικών δεδομένων. Για παράδειγμα, τα ομώνυμα, ομόφωνα, ο σαρκασμός, οι μεταφορές και οι εξαιρέσεις της γραμματικής είναι μερικές μόνο από τις ανωμαλίες της ανθρώπινης γλώσσας που οι άνθρωποι χρειάζονται αρκετή εξάσκηση για να τις κατανοήσουν πλήρως.

Για να είναι χρήσιμες οι εφαρμογές που κατανοούν τη φυσική γλώσσα πρέπει να είναι σε θέση από την αρχή να αντιλαμβάνονται αυτές τις ανωμαλίες και να αντιδρούν σωστά σε αυτές. Το NLP μέσω αρκετών διεργασιών αναλύει τα ανθρώπινα δεδομένα κειμένου και φωνής με τρόπους που βοηθά τον υπολογιστή να κατανοήσει τι προσλαμβάνει.

Ορισμένες από αυτές τις διεργασίες περιλαμβάνουν τα ακόλουθα:

- **Αναγνώριση ομιλίας (Speech Recognition***):* Είναι η μετατροπή φωνητικού μηνύματος (εισόδου) σε γραπτό λόγο (κείμενο), για αυτό και είναι γνωστή και ως ομιλία σε κείμενο.

- **Επισήμανση μερών του λόγου (Speech Tagging):** Είναι η απόφαση του σε ποιο γραμματικό μέρος ανήκει μία λέξη βάσει της πρότασης που την εμπεριέχει. Συνήθως, η πρόταση χωρίζεται σε πρόσωπο, ρήμα, ουσιαστικό και επίθετο.

- **Αποσαφήνιση έννοιας της λέξης (Word Sense Disambiguation):** Είναι η διαδικασία επιλογής της σημασίας μιας λέξης με πολλαπλές σημασίες μέσω σημασιολογικής ανάλυσης που καθορίζει τη λέξη που έχει περισσότερο νόημα στο δεδομένο πλαίσιο. Για παράδειγμα, στην πρόταση «Χτυπήθηκε από σφαίρα και μεταφέρθηκε στο νοσοκομείο.», η λέξη «σφαίρα» αναφέρεται στη σφαίρα από όπλο, ενώ στην πρόταση «Η Γη είναι υδρόγειος σφαίρα.» , η λέξη «σφαίρα» αναφέρεται στο στερεό σχήμα.

- **Εξαγωγή βασικών οντοτήτων (Named Entity Recognition or NEM):** Είναι η διαδικασία αναγνώρισης των βασικών οντοτήτων (πληροφοριών) από μια πρόταση όπως, ημερομηνίες, κύρια ονόματα, τηλέφωνα και τοποθεσίες. Για παράδειγμα στην πρόταση «Το τηλέφωνο μου είναι 12345678», το «12345678» είναι βασική οντότητα αφού είναι η κύρια πληροφορία που θέλει να μεταφέρει η πρόταση.

- **Ανάλυση συναισθήματος (Sentiment Analysis):** Είναι η διαδικασία αντίληψης του συναισθήματος του ομιλητή ή συγγραφέα. Συνήθως, κατά τη διαδικασία αυτή γίνεται προσπάθεια να αναγνωριστεί κατά πόσο το περιεχόμενο των δεδομένων είναι θετικό, αρνητικό ή ουδέτερο. Η πρόταση «Πονάω το κεφάλι μου.», είναι πρόταση η οποία περιέχει αρνητικό περιεχόμενο, η πρόταση «Πέρασα τέλεια στο χτεσινό τραπέζι.» περιέχει θετικό περιεχόμενο και η πρόταση «Θα πάω να πιάσω τα παιδιά από το σχολείο.», περιέχει ουδέτερο συναισθηματικά περιεχόμενο. Επιπρόσθετα, μέσω της τεχνικής αυτής γίνονται κατανοητά και συναισθήματα όπως χαρά, λύπη, θυμός και αναγνωρίζεται η ειρωνεία από την κυριολεξία [17].

3.1.4 Natural Language Understanding (NLU)

Το NLU (Natural Language Understanding – «Κατανόηση Φυσικής Γλώσσας») αποτελεί κομμάτι του NLP που επεξηγήθηκε πιο πάνω. Ασχολείται με την κατανόηση της μηχανικής ανάγνωσης, δηλαδή το πώς μια μηχανή διαβάζει και κατανοεί τη φυσική γλώσσα.

Οι αλγόριθμοι NLU επεξεργάζονται κείμενο χρησιμοποιώντας υπολογιστικές μεθόδους, με στόχο να γίνει κατανοητό το κείμενο εισόδου και να πραχθεί αποτέλεσμα. Πιο συγκεκριμένα, μετατρέπουν το κείμενο σε μορφή κατανοητή από τη μηχανή [32].

Γενικά οι μέθοδοι αυτοί μπορούν να χρησιμοποιηθούν με πολλούς διαφορετικούς τρόπους, συμπεριλαμβανομένης της κατανόησης του διαλόγου μεταξύ δύο ατόμων, της κατανόησης του πώς αισθάνεται κάποιος για μια συγκεκριμένη κατάσταση και άλλων παρόμοιων σεναρίων.

Υπάρχουν τρία γλωσσικά επίπεδα για την κατανόηση του NLU:

- **Σύνταξη:** Η διαδικασία γραμματικής ανάλυσης της πρότασης ώστε να διαπιστωθεί κατά πόσο η πρόταση είναι γραμματικά ορθή
- **Σημασιολογία:** Η διαδικασία ανάλυσης λεπτομερειών του κειμένου που προκύπτουν από συμφραζόμενα, όπως ο τόνος της φωνής και η επιλογή λέξεων.
- **Πραγματική ανάλυση:** Η διαδικασία κατανόησης του πλαισίου παραγωγής του κειμένου και του τι αυτό προσπαθεί να επιτύχει.

Βασικό στοιχείο του NLU είναι η προσπάθεια πλήρους γραμματικής και συναισθηματικής κατανόησης ενός κειμένου και όχι η απλή αναγνώριση λέξεων σε αυτό. Έτσι είναι άμεσα συνδεδεμένο και με την αλληλεπίδραση ανθρώπου υπολογιστή, αφού επιτρέπει στον υπολογιστή να επικοινωνήσει με τους ανθρώπους στη δική τους γλώσσα .

3.2 Λογισμικό ανάπτυξης

3.2.1 Έρευνα για επιλογή λογισμικού ανάπτυξης

Για τη δημιουργία ενός chatbot έχουν αναπτυχθεί αρκετά λογισμικά. Τέτοια λογισμικά χρησιμοποιούνται ώστε να δημιουργούνται πιο εύκολα και γρήγορα chatbots τα οποία είναι αποδοτικά και εύκολα στη χρήση. Όμως για να μπορεί κάποιος να είναι σε θέση να τα διαχειριστεί σωστά, θα πρέπει να είναι βασικός γνώστης των τεχνικών και των εργαλείων που χρησιμοποιεί το εκάστοτε λογισμικό.

Για την ανάπτυξη της παρούσας Ατομικής Διπλωματικής Εργασίας έγινε μια έρευνα ώστε να βρεθεί το κατάλληλο λογισμικό. Πιο κάτω αναφέρονται επιγραμματικά τρία από τα υποψήφια λογισμικά και παρουσιάζονται σε συντομία οι λόγοι που τελικά απορρίφθηκαν.

3.2.1.1 PyTorch

Το PyTorch είναι ένα open source machine learning framework, το οποίο χρησιμοποιείται για τη δημιουργία έξυπνων πρακτόρων βασισμένων στην Τεχνητή Νοημοσύνη και κυρίως στη Μηχανική Μάθηση. Το λογισμικό αυτό είναι σε θέση να βοηθήσει στη δημιουργία ενός chatbot ή κάποιου άλλου virtual assistant αφού αυτά αποτελούν έξυπνους πράκτορες [30].

Το PyTorch δεν χρησιμοποιήθηκε στην παρούσα Διπλωματική Εργασία για τον λόγο ότι αποτελεί ένα αρκετά γενικό framework αφού δεν επικεντρώνεται στη δημιουργία chatbot αλλά παρέχει γενικές δυνατότητες ικανές για τη δημιουργία ενός οποιουδήποτε έξυπνου πράκτορα.

3.2.1.2 DevPavlov

Το DevPavlov είναι ένα open source framework για τη δημιουργία chatbots και virtual assistants. Διαθέτει ολοκληρωμένα και ευέλικτα εργαλεία που επιτρέπουν σε προγραμματιστές και ερευνητές του πεδίου της Τεχνητής Νοημοσύνης να δημιουργήσουν πολύπλοκους βοηθούς συνομιλίας πολλαπλών δεξιοτήτων. Το λογισμικό αυτό βασίζεται στο PyTorch το οποίο παρουσιάζεται πιο πάνω[12].

Το συγκεκριμένο λογισμικό αν και φαινόταν αρκετά εύκολο στη χρήση και καλύπτει όλες τις απαραίτητες προδιαγραφές, εν τέλει δεν φάνηκε και ιδιαίτερα χρήσιμο στην περίπτωση του εν λόγω chatbot. Περισσότερες λεπτομέρειες για τις προδιαγραφές και απαιτήσεις που υπήρχαν για την ανάπτυξη του συγκεκριμένου chatbot αναλύονται στο επόμενο κεφάλαιο αυτού του δοκιμίου.

3.2.1.3 Rasa

Το Rasa είναι μια πλατφόρμα μετατροπής του τρόπου με τον οποίο αλληλεπιδρούν οι άνθρωποι με οργανισμούς μέσω συνομιλιών Τεχνητής Νοημοσύνης. Χρησιμοποιείται και αυτή για τη δημιουργία chatbots, τα οποία αποτελούν έξυπνους βοηθούς [31].

Το Rasa ήταν ιδανικό για το απαιτούμενο chatbot, όμως απορρίφθηκε λόγω οικονομικών περιορισμών. Πιο συγκεκριμένα, παρόλο που ήταν, μέχρι και πέρσι, μια open source πλατφόρμα, μετά την τελευταία αναβάθμισή της προστέθηκε ένα αρκετά υψηλό χρηματικό κόστος για τη χρήση της.

3.2.2 Λογισμικό Ανάπτυξης – BotPress

Μετά από έρευνα και αρκετές δοκιμές, αποφασίστηκε τελικά η χρήση του BotPress. Το BotPress αναπτύχθηκε το 2015 και είναι ένα open-source machine learning framework ειδικά διαμορφωμένο για τη δημιουργία chatbots. Το λογισμικό αυτό είναι αρκετά κατανοητό και εύκολο στη χρήση και επιτρέπει στους προγραμματιστές να χτίσουν ένα chatbot όπως ακριβώς το θέλουν [7].

Σχήμα 3.3 Λογότυπο του BotPress

Οι λόγοι επιλογής του BotPress σαν πλατφόρμα ανάπτυξής είναι οι ακόλουθοι:

- Δίνει τη δυνατότητα να δημιουργήσουμε με ευκολία chatbots με μεγάλη ακρίβεια, με αποτέλεσμα να δίνεται η αίσθηση συνομιλίας με κάποιο πραγματικό άνθρωπο.
- Είναι δωρεάν.
- Δίνει τη δυνατότητα να δημιουργήσουμε custom actions, τα οποία θα αναλυθούν σε μετέπειτα στάδιο, με αποτέλεσμα να είναι δυνατή η σύνδεση στη MySQL βάση δεδομένων.

- Επιτρέπει τη δημιουργία επιλογών, οι οποίες σε αυτή την περίπτωση είναι αρκετά βοηθητικές, αφού όπως θα δούμε και σε επόμενα κεφάλαια, ο χρήστης μπορεί να μην ξέρει ακριβώς τι ψάχνει ή τι ερώτηση πρέπει να υποβάλει.

Για την εγκατάσταση και χρήση του, μετά την τελευταία αναβάθμιση του, το BotPress, παρέχει στους χρήστες δύο επιλογές:

- Να κατεβάσουν από το GitHub τον κώδικα του botpress και να χτίσουν τοπικά τον δικό τους botpress server .
- Να εγγραφούν στο BotPress Cloud και να χτίσουν το chatbot τους σε κομμάτι sever που ανήκει στην εταιρεία.

Για τη δημιουργία του chatbot της παρούσας διπλωματικής εργασίας επιλέχθηκε η πρώτη επιλογή, δηλαδή έγινε λήψη των απαραίτητων αρχείων από το GitHub και χτίστηκε τοπικά server. Ο λόγος που προτιμήθηκε αυτή η επιλογή, παρόλο που το Botpress cloud επιτρέπει πολύ πιο εύκολα και γρήγορα τη δημοσίευση του chatbot στο web, είναι το γεγονός ότι δεν ήταν δυνατή η σύνδεση σε δική μας βάση, λόγω περιορισμών στα δικαιώματα.

Το BotPress αποτελείται από πολλά συστατικά μέρη. Τα πιο σημαντικά είναι τα Flows, το NLU engine, το Q&A, το Code Editor και ο Emulator. Καθένα από αυτά τα μέρη χρησιμοποιείται για να χτιστούν σωστά διάφορες λειτουργίες του chatbot. Επιγραμματικά τα Flows χρησιμοποιούνται για να χτιστούν τα μονοπάτια συζήτησης. Το NLU engine χρησιμοποιείται για να δηλωθούν τα intents και entities τα οποία βοηθούν στη σωστή κατανόηση του κειμένου εισόδου. Το Q&A χρησιμοποιείται για τη δημιουργία ερώτησης-απάντησης. Ο Code Editor χρησιμοποιείται σε περίπτωση που θέλουμε να χτίσουμε τυποποιημένες απαντήσεις για την κατασκευή custom actions τα οποία στο σύστημα που αναπτύσσεται στην παρούσα διπλωματική εργασία αποτελούν και το μέσο σύνδεσης με τη βάση δεδομένων. Τέλος, ο Emulator χρησιμοποιείται για τη δοκιμή του chatbot ανά πάσα χρονική στιγμή.

Ειδικότερα τα Flows, είναι ένα βασικό μέρος του botpress studio στο οποίο χτίζονται τα μονοπάτια συζήτησης, με τη χρήση nodes και transitions. Τα nodes αποτελούν ένα κόμβο σε μια συζήτηση. Όταν μεταβούμε σε κάποιο από αυτά, το node εκτελεί τις λειτουργίες που έχει, περιμένει για κάποια ενέργεια και ανάλογα με αυτήν αποφασίζει ποιο transition θα εκτελεστεί και οδηγούμαστε σε ένα άλλο node. Ένα node μπορεί να είναι και choice node. Στην περίπτωση αυτή, το chatbot δίνει στον χρήστη επιλογές και ο χρήστης καλείται να επιλέξει μια από αυτές. Ανάλογα με την επιλογή μεταβαίνουμε σε επόμενο κόμβο. Ένα node χωρίζεται σε τρία βασικά μέρη το OnEnter, OnRecieve και Transitions.

Το NLU (Natural Language Understanding) engine είναι, επίσης, ένα βασικό μέρος του botpress studio. Αποτελείται από δύο επιμέρους κομμάτια, τα Intents και Entities. Σε κάθε NLU engine τα intents απαντούν σε μια ενέργεια, ενώ τα entities απομονώνουν σημαντική πληροφορία από ένα κείμενο.

Στο κομμάτι που βρίσκονται τα intents δηλώνεται με κάποιο χαρακτηριστικό id κάθε intent που θέλουμε και σε αυτό τοποθετούμε όσες περισσότερες πιθανές περιπτώσεις κειμένου με ίδιο νόημα μπορούμε. Για παράδειγμα, ένα intent μπορεί να είναι η απάντηση σε περίπτωση που ο χρήστης ζητήσει βοήθεια. Άρα, δηλώνουμε ένα intent με id = help και γράφουμε όσους περισσότερους δυνατούς τρόπους μπορούμε να σκεφτούμε με τους οποίους ένας χρήστης θα ζητούσε βοήθεια, όπως “Help”, “I need help” και τα λοιπά.

Στο κομμάτι που βρίσκονται τα entities δηλώνεται πάλι με κάποιο χαρακτηριστικό id κάθε entity που θέλουμε. Σε αντίθεση με το intent, εδώ δηλώνουμε διάφορα κομμάτια κειμένου τα οποία πιθανόν να περιέχουν την πληροφορία που θέλουμε και δηλώνουμε τι σε αυτά τα κομμάτια είναι το entity που ψάχνουμε. Για παράδειγμα, αν θέλουμε να εντοπίσουμε την ημερομηνία από το κείμενο εισόδου, δηλώνουμε entity με id=date και γράφουμε πιθανά σενάρια, όπως «η ημερομηνία είναι 1/1/2001» ή «1 Γενάρη του 2001» και επιλέγω ποιο κομμάτι αναφέρεται σε κάποια ημερομηνία.

Όλα αυτά τα δεδομένα που κατασκευάζουμε αποτελούν είσοδο στον αλγόριθμο μηχανικής μάθησης που τρέχει πίσω από το chatbot. Έτσι, όταν επιλέξουμε να εκπαιδεύσουμε το chatbot μας, ουσιαστικά εκπαιδεύουμε τον αλγόριθμο με τα δεδομένα που κατασκευάσαμε ώστε σε ζωντανή συζήτηση το chatbot να μπορεί να αναγνωρίζει ικανοποιητικά τι λέει ο χρήστης.

Όσον αφορά το κομμάτι Q&A, είναι ένα μέρος του botpress studio που επιτρέπει τη δημιουργία τυποποιημένων ερωτήσεων και απαντήσεων. Δεν είναι απαραίτητη η χρήση του. Χρησιμοποιείται, συνήθως, σε περιπτώσεις που είναι πολύ συγκεκριμένες οι ερωτήσεις και έχουν μια πολύ συγκεκριμένη απάντηση. Ο τρόπος που λειτουργεί μοιάζει με αυτό των intents, αφού ουσιαστικά ακολουθεί την ίδια λογική. Δηλώνουμε με ένα μοναδικό id ένα Q&A και ακολούθως δηλώνουμε την ερώτηση με όσους περισσότερους διαφορετικούς τρόπους μπορούμε και την απάντηση.

Η χρήση του Q&A δεν είναι καλή τακτική σε περιπτώσεις που δεν θέλουμε να ξεφεύγουμε καθόλου από τη ροή συζήτησης, γιατί ουσιαστικά λειτουργεί σαν event και μας μετακινεί από τη ροή. Χρησιμοποιείται, κυρίως, σε περιπτώσεις που το chatbot που θέλουμε να δημιουργήσουμε δεν ακολουθεί κάποιο συγκεκριμένο μονοπάτι συζήτησης και απλά απαντά σε ανεξάρτητες μεταξύ τους ερωτήσεις. Στο chatbot που αναπτύσσεται στην παρούσα διπλωματική εργασία δεν χρησιμοποιήθηκε το Q&A αφού οι περισσότερες ερωτήσεις και απαντήσεις είναι αλληλένδετες μεταξύ τους και πρέπει να ακολουθούν συγκεκριμένη ροή συζήτησης.

Τέλος, ο Code Editor αποτελεί ένα αρκετά σημαντικό κομμάτι του botpress studio, αν και δεν είναι απαραίτητη η χρήση του για τη δημιουργία ενός chatbot. Παρόλα αυτά, στην παρούσα περίπτωση είναι απαραίτητο και αποτελεί ένα κομβικό σημείο, αφού επιτρέπει τη δημιουργία custom actions. Τα custom actions γράφονται σε γλώσσα προγραμματισμού JavaScript και σου δίνουν τη δυνατότητα να καθορίσεις εσύ τη δράση (action) που θα εκτελεστεί σε κάποιο node. Το παρόν chatbot, χρησιμοποιεί τα custom actions για να επικοινωνεί με τη βάση δεδομένων, να εκτελεί κάποιο query και να επιστρέφει στο node κάποια πληροφορία [6].

3.3 Απαιτούμενες Τεχνολογίες

Για να υλοποιηθεί το chatbot της παρούσας διπλωματικής εργασίας, χρησιμοποιήθηκαν, σε συνδυασμό με το botpress, και άλλες τεχνολογίες. Αρχικά, η βάση δεδομένων που χρησιμοποιείται είναι βάση MySQL και για την αναζήτηση των κατάλληλων δεδομένων σε αυτή, χρησιμοποιήθηκαν SQL queries. Όσον αφορά το botpress, χρησιμοποιεί JSON αρχεία, που έχουν αποθηκευμένη τη δομή του chatbot και JavaScript μέσω της οποίας επιτρέπεται η δημιουργία custom actions. Πιο κάτω αναλύονται οι τεχνολογίες αυτές και εξηγείται εν συντομία το πώς εμπλέκονται στην υλοποίηση του chatbot.

3.3.1 MySQL

Το MySQL είναι ένα από τα δημοφιλέστερα συστήματα διαχείρισης βάσεων δεδομένων ανοιχτού κώδικα που υπάρχουν σήμερα στον κόσμο. Η MySQL ανήκει στη Σουηδική εταιρεία MySQL AB, η οποία εξαγοράστηκε από τη Sun Microsystems, τώρα γνωστή ως Oracle Corporation. Το σύστημα αυτό, είναι σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και χρησιμοποιείται, κυρίως, για διαδικτυακά προγράμματα και ιστοσελίδες. Μια σχεσιακή βάση δεδομένων οργανώνει δεδομένα σε έναν ή περισσότερους πίνακες, στους οποίους τα δεδομένα μπορεί να σχετίζονται μεταξύ τους.

Το MySQL για τη διαχείριση των δεδομένων χρησιμοποιεί SQL. Η SQL είναι γλώσσα που χρησιμοποιείται για τη δημιουργία, την τροποποίηση και την εξαγωγή δεδομένων από τη σχεσιακή βάση δεδομένων, καθώς και τον έλεγχο της πρόσβασης των χρηστών στη βάση αυτή. Εκτός από τις σχεσιακές βάσεις δεδομένων και το SQL, ένα RDBMS (Relational DataBase Management System), όπως το MySQL, λειτουργεί με ένα λειτουργικό σύστημα για την υλοποίηση μιας σχεσιακής βάσης δεδομένων στο σύστημα αποθήκευσης ενός υπολογιστή, διαχειρίζεται τους χρήστες, επιτρέπει την πρόσβαση στο δίκτυο και διευκολύνει τον έλεγχο της ακεραιότητας της βάσης δεδομένων και τη δημιουργία αντιγράφων ασφαλείας [28].

3.3.2 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου και αποτελεί μια από τις βασικές τεχνολογίες του Παγκόσμιου Ιστού, μαζί με την HTML και το CSS. Από το 2022, το 98% των ιστοχώρων χρησιμοποιούν JavaScript στο frontend. Όλα τα μεγάλα προγράμματα περιήγησης ιστού διαθέτουν ειδική μηχανή JavaScript για την εκτέλεση του κώδικα στις συσκευές των χρηστών [27].

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, η Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, λόγω του ότι η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java.

Κυκλοφόρησε για πρώτη φορά στην αγορά τον Σεπτέμβριο του 1995 σε βήτα (beta) εκδόση με το πρόγραμμα Netscape Navigator εκδοχή 2.0, το οποίο ήταν πρόγραμμα περιήγησης στο Web, με το όνομα LiveScrit. Η LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου 1995, όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape εκδοχή 2.0B3 [24][15].

3.3.3 JSON files

Ένα JSON file είναι ένα αρχείο που αποθηκεύει απλές δομές δεδομένων και αντικείμενα σε μορφή JavaScript Object Notation (JSON). Η μορφή αυτή είναι μια τυπική μορφή ανταλλαγής δεδομένων και χρησιμοποιείται κυρίως για τη μετάδοση δεδομένων μεταξύ μιας εφαρμογής ιστού και ενός διακομιστή. Αν και βασίστηκε στα αρχικά στάδια σε ένα υποσύνολο της γλώσσας JavaScript, η μορφή JSON θεωρείται μια ανεξάρτητη μορφή και υποστηρίζεται από πολλά διαφορετικά API προγραμματισμού. Με την πάροδο του χρόνου γίνεται όλο και πιο δημοφιλής ως εναλλακτική της XML.

Δεδομένου ότι οι προγραμματιστές χρησιμοποιούν JSON για να υποστηρίξουν την ανάπτυξη λογισμικού, ορισμένοι μπορεί να πιστεύουν ότι τα αρχεία JSON είναι περίπλοκα και αρκετά δύσκολα στην κατανόηση, όμως αποτελούν μια πολύ απλή μέθοδο διαχείρισης δεδομένων. Γενικά, τα αρχεία JSON είναι βασισμένα σε κείμενο το οποίο είναι αρκετά εύκολα αναγνώσιμό από οποιοδήποτε άνθρωπο και εύκολα μπορεί κανείς να τα επεξεργαστεί με τη χρήση ενός προγράμματος επεξεργασίας κειμένου [29].

3.3.4 HTML/CSS

Η HTML και η CSS είναι δύο από τις βασικές τεχνολογίες για τη δημιουργία ιστοσελίδων. Τα ακρώνυμα της HTML είναι Hypertext Markup Language και του CSS είναι Cascading Style Sheets. Συνοπτικά, η HTML παρέχει τη δομή της σελίδας, ενώ το CSS την (οπτική και ακουστική) διάταξη της σελίδας. Οι δύο αυτές τεχνολογίες, μαζί με τα γραφικά και το scripting αποτελούν τη βάση για την κατασκευή ιστοσελίδων και εφαρμογών Ιστού.

Πιο συγκεκριμένα η HTML είναι ένα σύνολο κανόνων που διαμορφώνουν την εμφάνιση και το περιεχόμενο μιας ιστοσελίδας. Ουσιαστικά, δεν είναι γλώσσα προγραμματισμού, αλλά γλώσσα περιγραφής ιδιοτήτων των στοιχείων που αποτελούν μία ιστοσελίδα. Επίσης, μπορεί να γραφτεί κατευθείαν σε πηγαίο κώδικά ή να πραχτεί από κάποιο πρόγραμμα κατασκευής ιστοσελίδων [18].

Το CSS προστέθηκε πρώτη φορά στην HTML 4.0. Σκοπός του είναι να καθορίζει πως εμφανίζονται στον επισκέπτη μιας σελίδας τα διάφορα στοιχεία της HTML. Η προσθήκη αυτή έλυσε το πρόβλημα της μορφοποίησης των σελίδων, σώζοντας τους σχεδιαστές από πολύ κόπο και πολύ χρόνο, μειώνοντας σημαντικά τον όγκο της εργασίας [19].

Κεφάλαιο 4 – Ανάλυση Απαιτήσεων και Προδιαγραφές

4.1 Εισαγωγή

Ο κύκλος ζωής ενός λογισμικού αποτελείται από την καταγραφή των απαιτήσεων και προδιαγραφών, τον σχεδιασμό του, την υλοποίηση του, τον έλεγχο, την ενσωμάτωση του, τη λειτουργία του και τέλος τη συντήρηση του. Στο κεφάλαιο αυτό θα επικεντρωθούμε στην ανάλυση των βασικών απαιτήσεων και προδιαγραφών του συστήματος, έχοντας υπόψη πως οι απαιτήσεις πρέπει να διατυπώνονται με ακρίβεια και να καλύπτουν όλες τις επιθυμητές λειτουργίες.

Για τον καθαρισμό των απαιτήσεων και προδιαγραφών ενός λογισμικού χρησιμοποιείται η Μηχανική Απαιτήσεων. Μηχανική Απαιτήσεων ονομάζεται η διαδικασία για τον καθορισμό των επιθυμητών υπηρεσιών/λειτουργιών από ένα σύστημα και των περιορισμών που απαιτούνται για τη λειτουργία και την ανάπτυξή του [22].

4.2 Ανάλυση Απαιτήσεων

Οι απαιτήσεις χωρίζονται σε Λειτουργικές και Μη-Λειτουργικές. Οι Λειτουργικές απαιτήσεις απαντούν στις ερωτήσεις «Τι πρέπει να κάνει το σύστημα;», «Πώς πρέπει να αντιδρά στις διάφορες εισόδους;» και «Πώς πρέπει να συμπεριφέρεται σε συγκεκριμένες καταστάσεις;». Οι Μη-Λειτουργικές απαιτήσεις αναφέρονται στο σύστημα ως σύνολο και σχετίζονται με τους περιορισμούς που υπάρχουν σε σχέση με το περιβάλλον ανάπτυξης, τη γλώσσα προγραμματισμού και τη μέθοδο ανάπτυξης ,όπως περιορισμούς σε απόδοση, αξιοπιστία και αποθηκευτικό χώρο.

Για τη συλλογή των απαιτήσεων έγινε καταγραφή παραδειγμάτων συνομιλίας ενός χρήστη με το chatbot, καθώς επίσης, και έρευνα σε σχέση με το τι περιέχουν τα έγγραφα Patient Summary και e-Prescription. Στη διαδικασία αυτή αναλύθηκαν τα ακριβή δεδομένα που περιέχονται στα ιατρικά αυτά έγγραφα και το πώς θα πρέπει να εμφανίζονται στον χρήστη και χωρίστηκε η διαδικασία σε μέρη ανάλογα με την ερώτηση του χρήστη προς στο σύστημα.

4.2.1 Λειτουργικές Απαιτήσεις

Το παρόν chatbot, όπως ήδη αναφέρθηκε, σκοπό έχει την παροχή πληροφοριών στον χρήστη που σχετίζονται με το προσωπικό του Patient Summary και e-Prescription. Ένας χρήστης θα πρέπει να μπορεί να εισέρχεται στο σύστημα και να υποβάλλει την ερώτηση που αυτός επιθυμεί. Ανάλογα με την ερώτηση που υποβάλλεται, το chatbot θα πρέπει να είναι σε θέση να αναγνωρίζει τι πληροφορία πρέπει να ψάξει, να τη βρίσκει και να την επιστρέφει σαν απάντηση στον χρήστη.

Έτσι οι λειτουργικές απαιτήσεις που έχει το σύστημα είναι οι ακόλουθες:

- **Είσοδος:** Ο χρήστης πρέπει να μπορεί να εισέρχεται στο σύστημα και να καταχωρεί την ταυτότητά του, ώστε να γίνεται ταυτοποίηση του και να γνωρίζει το chatbot με ποιον συνομιλεί.
- **Παροχή βοήθειας:** Το chatbot πρέπει να μπορεί να παρέχει βοήθεια στον χρήστη οποιαδήποτε στιγμή της ροής συζήτησης και να τον κατευθύνει να βρει αυτό που ψάχνει εύκολα και γρήγορα.
- **Ερωτήσεις:** Το chatbot πρέπει να μπορεί να απαντήσει σε όλες τις ερωτήσεις σχετικά με το Patient Summary και e-Prescription. Στην παρούσα ατομική διπλωματική εργασία το σύστημα που αναπτύχθηκε είναι σε θέση να απαντήσει σε γενικές μόνο ερωτήσεις. Περισσότερα για τις ερωτήσεις επεξηγούνται πιο κάτω.
- **Έξοδος:** Το chatbot πρέπει να τερματίζει τη ροή συζήτησης σε περίπτωση που ο χρήστης ζητήσει να αποσυνδεθεί.

Όσον αφορά στις ερωτήσεις που μπορεί να απαντήσει το chatbot σε σχέση με το Patient Summary του χρήστη, αυτές έχουν ομαδοποιηθεί σε κατηγορίες ανάλογα με την πληροφορία που ζητούν. Οι κατηγορίες είναι οι ακόλουθες:

- **Αλλεργίες** - Σε αυτή την περίπτωση, το chatbot απαντά στον χρήστη πόσες αλλεργίες έχει και του δίνει τη δυνατότητα να μάθει περισσότερες πληροφορίες για συγκεκριμένη κατηγορία αλλεργιών.
  Πιθανές ερωτήσεις που μπορεί να υποβάλει ο χρήστης είναι: «What are my allergies?», «Do I have any allergies?», «Can you tell me my allergies?», «I want to know my allergies.». Το chatbot απαντά στον χρήστη πόσες αλλεργίες έχει, για παράδειγμα απαντά «You have 3 allergies», και του δίνει τη δυνατότητα να διαλέξει από ένα drop down list για ποια κατηγορία αλλεργιών θέλει να μάθει πληροφορίες. Μερικές από τις κατηγορίες που περιλαμβάνονται στη λίστα είναι: food allergies, drug allergies, allergy to substance κτλ. Ανάλογα με το τι επιλέγει ο χρήστης, επιστρέφεται σε αυτόν μια λίστα που αναγράφει όλες τις αλλεργίες του χρήστη που ανήκουν σε αυτή την κατηγορία, διευκρινίζοντας τι τις προκαλεί (π.χ. μπανάνα) και ποια είναι η αντίδραση του οργανισμού σε κάθε περίπτωση (π.χ. άσθμα). Παραδείγματος χάριν, αν ο χρήστης επιλέξει να μάθει πληροφορίες για το «Food allergies» το chatbot απαντά «You have: 1) Allergy in banana (Reaction: asthma) 2) Allergy in tomato (Reaction: Cardiac arrhythmia)».

- **Εμβολιασμοί** - Σε αυτή την περίπτωση, το chatbot λέει στον χρήστη το ιστορικό εμβολιασμού του, δηλαδή τι εμβόλια έχει βάλει και πότε. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι: «Tell me my vaccinations», «I want to know my vaccination history». Η απάντηση που θα δώσει το chatbot είναι μια λίστα, για παράδειγμα « 1) Cholera vaccine (Date: 2018-03-12) 2) Diphtheria + pertussis + tetanus vaccine (Date: 2010-02-05)».

- **Παλιές ασθένειες** - Σε αυτή την περίπτωση το chatbot, λέει στον χρήστη πόσες ασθένειες βρίσκονται καταχωρημένες στο ιατρικό ιστορικό του και αν ζητηθεί ενημερώνει τον χρήστη για το τι ασθένειες είναι, πόσο διάρκεσαν και τι θεραπεία ακολουθήθηκε. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι: «Can i see my illnesses history?», «Do I have any past illnesses?», «Show me my past diseases». Η απάντηση που θα δώσει το chatbot είναι για παράδειγμα «You have 5 illnesses in your history.» και θα δώσει την επιλογή στον χρήστη να ζητήσει να δει όλες τις πληροφορίες για αυτές. Πιο συγκεκριμένα, αν ο χρήστης επιλέξει να μάθει περισσότερες πληροφορίες το chatbot θα επιστρέψει σαν απάντηση μια λίστα με όλες τις παλιές ασθένειες του χρήστη. Παραδείγματος χάριν « 1) Influenza, virus not identified (Duration: 2016-09-12 to 2016-09-13) Resolution Circumstances: None 2) Abdominal aortic aneurysm, without mention of rupture (Duration: 2005-01-01to ongoing) Resolution Circumstances: None».

- **Επεμβάσεις** - Σε αυτή την περίπτωση, το chatbot λέει στον χρήστη πόσες επεμβάσεις έχει κάνει και αν ζητηθεί δίνει την περιγραφή της κάθε επέμβασης, καθώς επίσης και την ημερομηνία που αυτή υλοποιήθηκε. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι: «Show me my surgical procedures», «Can I see my surgeries?». Η απάντηση που θα δώσει το chatbot είναι «You have done 3 surgical procedures.» και θα δώσει την δυνατότητα στον χρήστη να επιλέξει να μάθει περισσότερες πληροφορίες. Αν ο χρήστης επιλέξει να μάθει περισσότερα τότε το chatbot θα απαντήσει με μια λίστα που θα περιέχει όλες τις επεμβάσεις. Παραδείγματος χάριν, « 1) Arthroscopy of knee with meniscus repair (Date: 2003-01-01) 2) Biliary tract endoscopy (Date: 2009-12-01) 3) Implantation to cardiovascular system (Date: 2015-10-06)».

- **Τρέχοντα Προβλήματα** - Σε αυτή την περίπτωση, το chatbot λέει στον χρήστη τα τρέχοντα προβλήματα. Τρέχοντα προβλήματα αποτελούν οι αλλεργίες, οι ασθένειες που εξακολουθούν να υπάρχουν, όπως για παράδειγμα χρόνιες παθήσεις ή κάποια ασθένεια που περνά τώρα ο χρήστης, οι ασθένειες που πέρασε ο ασθενής εντός των τελευταίων έξι μηνών και τυχόν εξαρτήσεις (π.χ. εξάρτηση στο αλκοόλ). Για να είναι πιο εύκολη η ανάγνωση όλων αυτών, το chatbot, αν ερωτηθεί για τα τρέχοντα προβλήματα εμφανίζει στον χρήστη ένα dropdown list το οποίο περιέχει τις κατηγορίες που αναφέρθηκαν και καλεί τον χρήστη να διαλέξει τι από τα πιο πάνω θέλει. Αν επιλέξει να δει τα allergies του τότε το chatbot απαντά όπως ακριβώς γίνεται στην κατηγορία Αλλεργίες που επεξηγήθηκε πιο πάνω. Αν επιλέξει ongoing illnesses, τότε εμφανίζει τις ασθένειες που ακόμα αντιμετωπίζει ο χρήστης, αν επιλέξει τις last 6 months illnesses, τότε του επιστρέφει μια λίστα με όλες τις ασθένειες που πέρασε εντός των τελευταίων έξι μηνών και τέλος αν επιλέξει social problems (εξαρτήσεις), τότε το chatbot επιστρέφει λίστα με τις εξαρτήσεις του χρήστη. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι «What are my current problems?», «Do I have any current problem?». Η απάντηση που θα δώσει το chatbot είναι «I want to know about…» και ένα dropdown list με τις κατηγορίες που αναφέρθηκαν.

- **Ιατρικές συσκευές και εμφυτεύματα** - Σε αυτή την περίπτωση, το chatbot λέει στον χρήστη πόσες ιατρικές συσκευές και εμφυτεύματα έχει και ακολούθως ο χρήστης μπορεί να μάθει περισσότερες πληροφορίες είτε για τα εμφυτεύματα είτε για τις ιατρικές συσκευές διαλέγοντας την κατάλληλη επιλογή από το dropdown list που εμφανίζει το chatbot. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι «Do I have any medical devices or implants?», «I want to know if I have any implants?», «Can you please tell me my implants?». Η απάντηση που θα δώσει το

chatbot είναι «You have 5 medical devices and/or implants» και θα δώσει στον χρήστη μια λίστα με τα medical devices και μια λίστα με τα implants (οι λίστες περιέχουν την ονομασία του εν λόγω εμφυτεύματος ή συσκευής και την ημερομηνία τοποθέτησης τους). Για παράδειγμα «Implants: 1) Ankle joint implant (Date: 2017-09-05) 2) Body wall implant (Date: 1999-01-01)».

- **Περίληψη Φαρμακευτικής Αγωγής** - Σε αυτή την περίπτωση, το chatbot δίνει στον χρήστη μια σύντομη περίληψη του e-Prescription του, δηλαδή της φαρμακευτικής αγωγής του. Πιθανές ερωτήσεις που μπορεί να υποβάλλει ο χρήστης είναι «Tell me my prescription summary», «Show me my e-Prescription summary». Η απάντηση που θα δώσει το chatbot είναι μια λίστα με τις φαρμακευτικές συνταγές που έχει στο ιστορικό του ο χρήστης δίνοντας την ουσία του φαρμάκου, τη μορφή του, την εταιρία παραγωγής του, τη δοσολογία, τη συχνότητα δοσοληψίας και τη διάρκεια πρόσληψης του, καθώς επίσης και τις οδηγίες που είχε καταχωρήσει ο γιατρός για τον ασθενή και την ημερομηνία που έγινε καταχώρηση της συνταγής από τον γιατρό. Για παράδειγμα «1) sodium Mon fluorophosphate Form: Bath additive Brand: Pfizer Unit dose: 2 Frequency: 2 per day Duration: 12 per day Instructions: Take it after food. One in the morning and one at night (Date of issue: 2019-04-04)».

Τέλος, λόγω του ότι σε περιπτώσεις έκτακτης ανάγκης, χειρουργείου και τα λοιπά, ο γιατρός χρειάζεται άμεσα και γρήγορα συγκεκριμένες πληροφορίες, δόθηκε στο chatbot η δυνατότητα να τις εντοπίζει ακόμα γρηγορότερα όταν του δίνονται ταυτόχρονα παραπάνω από μια λέξεις κλειδιά.

Συγκεκριμένα στην κατηγορία Αλλεργίες αντί να ακολουθηθεί η διαδικασία που περιγράφεται πιο πάνω, ο χρήστης έχει τη δυνατότητα να ζητήσει να μάθει κατευθείαν συγκεκριμένες αλλεργίες ή δυσανεξίες που έχει σε φάρμακα, τρόφιμα ή ουσίες. Πληκτρολογώντας, για παράδειγμα, «I want to know my drug allergies.», το chatbot εμφανίζει αμέσως μια λίστα με όλα τα drug allergies που έχει ο χρήστης αντί να ακολουθήσει τη διαδικασία που περιγράφεται πιο πάνω.

4.2.2 Μη-Λειτουργικές Απαιτήσεις

Οι Μη-Λειτουργικές Απαιτήσεις, όπως αναφέρθηκε και πιο πάνω αφορούν στο σύστημα ως σύνολο και σχετίζονται με τους περιορισμούς που υπάρχουν σε σχέση με το περιβάλλον ανάπτυξης, τη γλώσσα προγραμματισμού και τη μέθοδο ανάπτυξης ,όπως περιορισμούς σε απόδοση, αξιοπιστία και αποθηκευτικό χώρο.

- **Απαιτήσεις Προϊόντος**
  Το chatbot για να τεθεί σε λειτουργία και να μπορεί να χρησιμοποιηθεί χρειάζεται να δημιουργηθεί ένα εικονικό περιβάλλον (virtual environment) το οποίο θα περιέχει ένα BotPress server. Για να γίνει αυτό, είναι αρκετό να κατεβάσει κανείς τα αρχεία από το GitHub directory και να τρέξει το .exe file (GitHub: https://github.com/botpress/botpress). Ακολούθως, εισάγει το chatbot στο admin panel και έτσι είναι σε θέσει να το τρέξει.

- **Απαιτήσεις Λειτουργικότητας**
  Για να τρέχει σωστά το σύστημα πρέπει να προσδιοριστούν τα συστήματα τα οποία συνεργάζονται για τη σωστή λειτουργία του. Για παράδειγμα, η βάση δεδομένων από την οποία αντλούνται τα δεδομένα.

- **Απαιτήσεις Απόδοσης**
  Θα πρέπει να ληφθεί υπόψη η αξιοπιστία των απαντήσεων που δίνει το chatbot, καθώς επίσης, και ο χρόνος απόκρισης της απάντησης.

4.3 Προδιαγραφές

4.3.1 Χαρακτηριστικά Χρήστη

Οι χρήστες του συστήματος χωρίζονται στις πιο κάτω κατηγορίες:

1. **Ασθενής**: Ασθενή αποτελεί οποιοδήποτε άτομο είναι καταχωρημένο στο σύστημα και έχει καταχωρημένο ιατρικό ιστορικό. Το chatbot χτίστηκε ειδικά για αυτή την κατηγορία χρηστών, αφού στοχεύει στο να μπορούν οι ασθενείς να βρίσκουν εύκολα τα ιατρικά δεδομένα τους.

2. **Γιατρός**: Γιατρό αποτελεί οποιοσδήποτε πάροχος υγείας είναι καταχωρημένος στο σύστημα. Το παρόν chatbot δεν έχει χτιστεί με γνώμονα αυτή την κατηγορία, αλλά μελλοντικός στόχος είναι να διαχωριστεί η κατηγορία αυτή και να προσφέρονται διαφορετικές λειτουργίες. Παρόλα αυτά, οι γιατροί μπορούν να εισέρχονται στο σύστημα, μόνο αν ο ασθενής τους δώσει τον αριθμό ταυτότητας του.

3. **Διαχειριστής**: Ο διαχειριστής του συστήματος έχει πλήρη πρόσβαση σε ολόκληρο το σύστημα και τα δεδομένα του. Μπορεί να προσθέτει, να αφαιρεί ή να τροποποιεί λειτουργίες που παρέχει το chatbot προσθέτοντας έτσι επιπλέον δυνατότητες στο σύστημα.

4.3.2 Αρχές για αποτελεσματική Σχεδίαση chatbot

Για τον αποτελεσματικό σχεδιασμό οποιουδήποτε chatbot υπάρχουν βασικές αρχές, οι οποίες στόχο έχουν τη δημιουργία ενός αποτελεσματικού εικονικού συνομιλητή που είναι σε θέση να βοηθήσει κάποιο άτομο δίνοντας του απαντήσεις σε ερωτήσεις που του θέτει. Ανάλογα με τον σκοπό για τον οποίο χτίζεται ένα chatbot είναι σε θέση να απαντήσει σε ένα συγκεκριμένο σύνολο ερωτήσεων.

Αρχικά, βασική και πρωταρχική αρχή αποτελεί ο καθορισμός του σκοπού με σαφήνεια. Είναι σημαντικό πριν ξεκινήσει η υλοποίηση ενός chatbot να οριστεί ξεκάθαρα ο σκοπός του και να διασφαλιστεί ότι ο σκοπός αυτός καλύπτει και ικανοποιεί τις ανάγκες του χρήστη. Αυτό περιλαμβάνει τον προσδιορισμό της βασικής λειτουργικότητας του chatbot, δηλαδή «Τι θα είναι σε θέση να κάνει;», «Τι ερωτήσεις θα μπορεί να απαντήσει;», του κοινού-στόχου, δηλαδή αν ο στόχος του chatbot καλύπτει τους στόχους/ανάγκες που έχει ένας χρήστης ο οποίος θα το χρησιμοποιήσει, και των αναμενόμενων αποτελεσμάτων.

Επίσης, βασικό στη δημιουργία ενός bot είναι να το κρατάμε απλό και κατανοητό. Τα Chatbots πρέπει να σχεδιάζονται με γνώμονα την απλότητα, αφού κύριος σκοπός κατασκευής ενός τέτοιου έξυπνου πράκτορα είναι να παρέχει βοήθεια και γενικά πληροφορίες εύκολα και γρήγορα. Αυτό σημαίνει ότι πρέπει να αποφεύγονται οι «συντριπτικοί» χρήστες, δηλαδή οι χρήστες οι οποίοι έχουν πολλές επιλογές ή πολύπλοκα μενού και να γίνεται χρήση συνοπτικής γλώσσας που είναι εύκολα κατανοητή. Είναι πολύ σημαντικό οποιοδήποτε chatbot να είναι εξειδικευμένο για συγκεκριμένα πράγματα (πεδίο ερωτήσεων), ώστε να είναι

ξεκάθαρη η δομή και ο στόχος του, αποφεύγοντας έτσι περίπλοκα μονοπάτια συζήτησης τα οποία μπορεί να οδηγήσουν το bot σε λανθασμένες απαντήσεις.

Επιπρόσθετα, εξίσου σημαντική είναι η εξατομίκευση της εμπειρίας χρήστη. Η εξατομίκευση μπορεί να βοηθήσει στη δημιουργία μιας καλύτερης εμπειρίας χρήστη. Αυτό μπορεί να περιλαμβάνει τη χρήση του ονόματος του χρήστη, την ανάμνηση προηγούμενων αλληλεπιδράσεων και την προσφορά σχετικών προτάσεων.

Τέλος, απαραίτητη είναι η παροχή καθαρών σημείων εξόδου καθώς και σημείων βοήθειας. Είναι σημαντικό να παρέχονται στους χρήστες σαφή σημεία εξόδου σε περίπτωση που θέλουν να εγκαταλείψουν το chatbot- όπως για παράδειγμα κουμπί «έξοδος» ή η παροχή σαφών οδηγιών για το πώς να τερματιστεί η συνομιλία- καθώς επίσης και σημείο αναφοράς/ βοήθειας- δηλαδή να παρέχεται στον χρήστη η δυνατότητα να ζητήσει βοήθεια ώστε να καταλάβει το πώς χρησιμοποιείται το chatbot, μπορώντας έτσι να έχει μια ολοκληρωμένη και εύκολη εμπειρία.

Συνολικά, ο αποτελεσματικός σχεδιασμός chatbot απαιτεί εστίαση στον χρήστη, σαφή επικοινωνία και δυνατότητα εξατομίκευσης της εμπειρίας. Ακολουθώντας αυτές τις αρχές, μπορεί να δημιουργηθεί ένα chatbot που είναι ελκυστικό, αποτελεσματικό και ευχάριστο [9][36].

Για την παρούσα Διπλωματική εργασία, ακολουθήθηκαν με προσοχή όλα όσα αναφέρθηκαν πιο πάνω. Αρχικά, ορίστηκε με σαφήνεια ποιος είναι ο σκοπός και στόχος αυτού του συστήματος και αναλύθηκαν με λεπτομέρεια ποιες είναι οι ανάγκες ενός πιθανού χρήστη, τι πληροφορίες πρέπει να παρέχονται και πώς πρέπει αυτές να εμφανίζονται. Ο σκοπός και οι ανάγκες του εν λόγω chatbot αναλύθηκαν και επεξηγήθηκαν σε προηγούμενα κεφάλαια και υποκεφάλαια του δοκιμίου.

Προχωρώντας, χτίστηκαν μονοπάτια συζήτησης απλά και κατανοητά, ώστε όλοι οι χρήστες να μπορούν να κατανοήσουν πλήρως τη λειτουργικότητα του chatbot και να είναι σε θέση να ρωτήσουν αυτό που θέλουν, παίρνοντας γρήγορα τη σωστή απάντηση. Αυτό επιτεύχθηκε, με την προσθήκη μονοπατιού βοήθειας. Όταν ο χρήστης μπερδευτεί ή δεν μπορεί να συνεχίσει τη συζήτηση, μπορεί να ζητήσει βοήθεια πληκτρολογώντας, παραδείγματος χάριν, «help» ή «I need help». Τότε, το chatbot του εξηγεί τον σκοπό και τον τρόπο λειτουργίας του, εξηγώντας παράλληλα και το πώς μπορεί ανά πάσα στιγμή να αποσυνδεθεί από το σύστημα ο χρήστης πληκτρολογώντας «logout». Επιπρόσθετα, του εμφανίζει μια λίστα με όλα τα μονοπάτια συζήτησης. Πιο συγκεκριμένα, η λίστα αυτή εμφανίζεται στον χρήστη σαν dropdown list και του επιτρέπει να διαλέξει ποια κατηγορία του patient summary θέλει να δει ή του e-Prescription.

Ανάλογα dropdown lists, περιέχονται σε διάφορα σημεία του chatbot, ώστε να βοηθούν τον χρήστη να εντοπίζει γρηγορότερα αυτό που ψάχνει. Για παράδειγμα, αν ο χρήστης ζητήσει να δει τις αλλεργίες του, τότε το chatbot του επιστρέφει μια λίστα που χωρίζει τις αλλεργίες σε κατηγορίες επιτρέποντας του να διαλέξει να δει πληροφορίες για συγκεκριμένη κατηγορία.

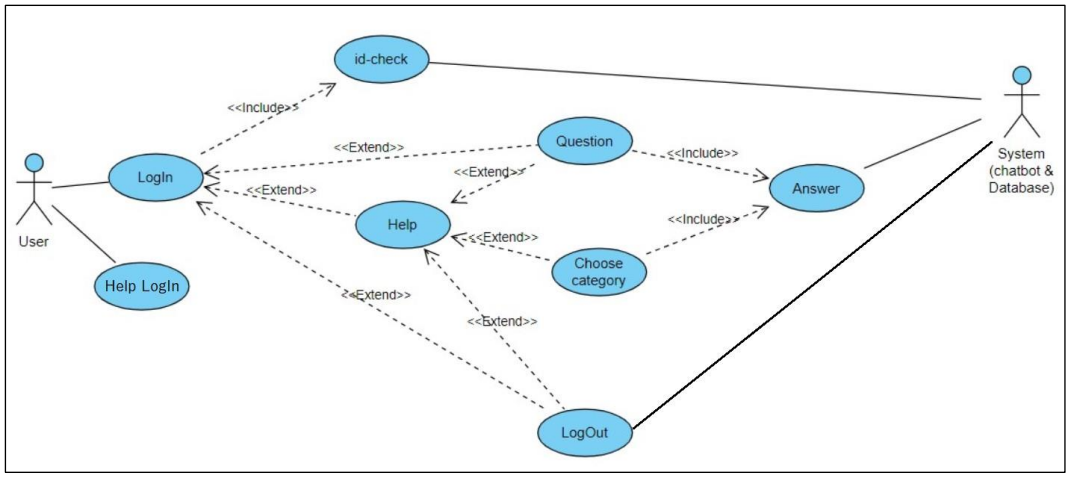
Επιπρόσθετα, το chatbot που αναπτύχθηκε προσφέρει μια πιο «προσωπική» εμπειρία αφού όταν ο χρήστης δίνει την ταυτότητα του, το σύστημα είναι σε θέση να αναγνωρίζει με ποιο άτομο συνομιλεί, και τον προσφωνεί με το όνομά του, για παράδειγμα «Hello Maria!». Τέλος, για να δίνεται η αίσθηση πραγματικής συνομιλίας, όταν το chatbot ψάχνει για την απάντηση και γράφει το μήνυμα εμφανίζεται στον χρήστη εικονίδιο που δείχνει ότι ο εικονικός βοηθός πληκτρολογεί.

4.3.3 Περιορισμοί σχεδιασμού

Για να έχει ένας χρήστης πρόσβαση στο chatbot, το μόνο που χρειάζεται είναι να έχει πρόσβαση στο διαδίκτυο, αφού το σύστημα είναι ενσωματωμένο σε μια ιστοσελίδα, συμβατή με όλους τους browsers. Η ιστοσελίδα αυτή δημιουργήθηκε μόνο για σκοπούς εξέτασης του chatbot. Σε κατοπινό στάδιο και αφού το chatbot περάσει τη φάση εξέτασης, θα ενσωματωθεί στο ήδη υπάρχον σύστημα. Για πρόσβαση στο σύστημα αυτό, η μόνη απαραίτητη προϋπόθεση θα είναι και πάλι η πρόσβαση στο διαδίκτυο.

4.3.4 User case diagram

Πιο κάτω βρίσκεται το user case diagram που δημιουργήθηκε για το παρόν σύστημα. Η δημιουργία ενός user case diagram βοηθά στο να ξεκαθαρίσουν οι αλληλεπιδράσεις που θα έχει ο χρήστης με το σύστημα. Πιο κάτω, για σκοπούς απλότητας δεν χωρίστηκαν οι κατηγορίες των ερωτήσεων, αλλά θεωρούμε πως για κάθε ερώτηση που απευθύνει ο χρήστης είμαστε στον κόμβο «Question».



Κεφάλαιο 5 – Σχεδιασμός Συστήματος/Υλοποίηση

5.1 Εισαγωγή

Σύμφωνα με τα στάδια ανάπτυξης λογισμικού, μετά την καταγραφή και ανάλυση των απαιτήσεων, ακολουθεί ο σχεδιασμός και η υλοποίηση του συστήματος. Ο σχεδιασμός ενός συστήματος περιλαμβάνει την καταγραφή και πλήρη περιγραφή των επιμέρους τμημάτων του συστήματος καθώς και των μεταξύ τους σχέσεων.

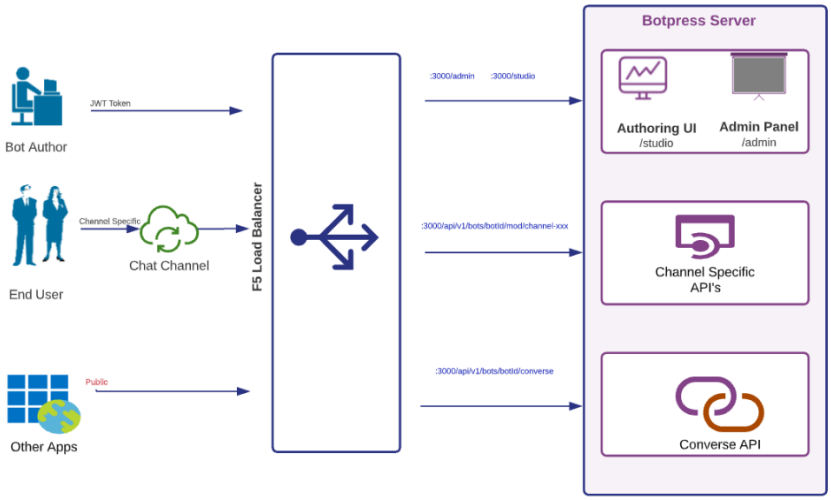
Ξεκινώντας τον σχεδιασμό του συστήματος της παρούσας Ατομικής Διπλωματικής Εργασίας, δημιουργήθηκε το user case diagram που παρουσιάστηκε στο προηγούμενο κεφάλαιο και στήθηκαν παραδείγματα συζητήσεων χρήστη και chatbot (κομμάτια συζητήσεων παρουσιάστηκαν σε προηγούμενα κεφάλαια και αναλύθηκαν στο Κεφάλαιο 4 στις λειτουργικές απαιτήσεις). Όσον αφορά, στην υλοποίηση του συστήματος, περιλαμβάνει τον σχεδιασμό του σε υπαρκτό υπολογιστικό σύστημα.

Στα υποκεφάλαια που ακολουθούν, περιγράφονται και παραθέτονται στιγμιότυπα από πιθανές συζητήσεις, στιγμιότυπα οθόνης και γίνεται πλήρης περιγραφή της πλατφόρμας που δημιουργήθηκε το chatbot (BotPress).

5.2 Γενική Αρχιτεκτονική του BotPress

Ο BotPress server περιέχει τρεις βασικές διεπαφές. Αυτή που αλληλεπιδρά ο δημιουργός του chatbot και αυτές που αλληλοεπιδρούν οι χρήστες, είτε κατευθείαν στο specific channel είτε μέσω κάποιου άλλου καναλιού (app).

Η διεπαφή που αλληλεπιδρά ο δημιουργός του chatbot χωρίζεται στο Admin Panel και το Studio (Authoring UI), τα οποία παρέχουν στον κατασκευαστή ένα ολοκληρωμένο περιβάλλον για την εύκολη δημιουργία ενός bot. Η διεπαφή που αλληλεπιδρούν οι χρήστες είναι ουσιαστικά κάποιο κανάλι που δημιουργήθηκε από τον κατασκευαστή ή κάποιο άλλο έτοιμο κανάλι (Messenger etc) [5].



Σχήμα 5.1 BotPress Interfaces Overview [5]

5.3 Σχεδιασμός και Υλοποίηση στο BotPress

5.3.1 Επισκόπηση

Η αρχιτεκτονική λογισμικού ενός βασικού chatbot αποτελείται από τρία βασικά μέρη:

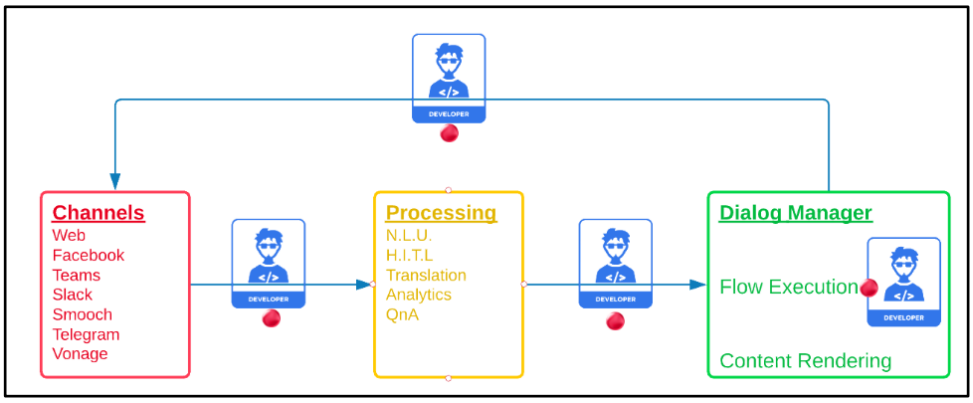
- **Κανάλια (Channels):** Ο τρόπος με τον οποίο το chatbot παίρνει τα μηνύματα.
- **Επεξεργασία (Processing):** Ο τρόπος με τον οποίο το chatbot επεξεργάζεται αυτά τα μηνύματα για να τα κατανοήσει και να τα μεταφράσει. Για την επεξεργασία χρησιμοποιούνται Analytics, NLU κ.τ.λ.
- **Απόφαση (Decision):** Ο τρόπος με τον οποίο το chatbot αποφασίζει ποια θα είναι η απάντηση που θα επιστρέψει στον χρήστη βάσει στοιχείων που βρίσκει ή έτοιμων αποφάσεων.

Συγκεκριμένα, στην πλατφόρμα Botpress, η οποία χρησιμοποιήθηκε για την ανάπτυξη του chatbot αυτής της διπλωματικής εργασίας, τα πιο πάνω μέρη υλοποιούνται ως εξής:

- **Channels:** Το Botpress δίνει στο chatbot τη δυνατότητα να στέλνει και να λαμβάνει μηνύματα μέσω μιας συγκεκριμένης πλατφόρμας συνομιλίας. Σε αντίθεση με τις περισσότερες πλατφόρμες στο BotPress τα κανάλια εγκαθίστανται και διαμορφώνονται μεμονωμένα και τοπικά, έτσι ο προγραμματιστής έχει τον πλήρη έλεγχο των δεδομένων που μεταδίδονται μεταξύ του bot και των πλατφόρμων συνομιλίας. Στα παρασκήνια (backend), το Botpress εφαρμόζει έναν μηχανισμό ουράς που επεξεργάζεται τα εισερχόμενα και τα εξερχόμενα μηνύματα διαδοχικά. Εάν ένα μήνυμα αποτύχει να υποβληθεί σε επεξεργασία ή να σταλεί, το μήνυμα θα επαναληφθεί πριν εμφανιστεί ένα μήνυμα σφάλματος στον προγραμματιστή και στο διαχειριστή του chatbot[5].

- **Processing (NLU):** Για την επεξεργασία των μηνυμάτων, το Botpress χρησιμοποιεί Κατανόηση Φυσικής Γλώσσας (ή NLU) η οποία επιτρέπει στο bot να επεξεργάζεται τα μηνύματα που λαμβάνονται από τις πλατφόρμες συνομιλίας, που είναι καθαρό μη δομημένο κείμενο, και να τα μετατρέπει σε δομημένα δεδομένα τα οποία μπορεί να κατανοήσει το bot. Οι κύριες εργασίες που κάνει ο κινητήρας NLU του BotPress είναι:
    1. **Intent Recognition:** αναγνώριση του τι θέλει ο χρήστης.
    2. **Entity Extraction:** εξαγωγή δομημένων πληροφοριών από μηνύματα όπως ημερομηνίες, ώρα, πόλεις, ονόματα και άλλα.
    3. **Slot tagging:** προσδιορισμός απαραίτητων παραμέτρων για την εκπλήρωση της συγκεκριμένης εργασίας.
    4. **Language identification:** αναγνώριση της γλώσσας που γράφει ο χρήστης και δόμηση του μηνύματος απάντησης στην ίδια γλώσσα [5].

- **Decision (Dialog Manager):** Το Botpress χρησιμοποιεί τον Διαχειριστή Διαλόγου (ή DM) για τη λήψη αποφάσεων. Ο ρόλος του DM είναι να καθορίσει τι πρέπει να κάνει ή να πει το bot στη συνέχεια. Παρόλο που ο Διαχειριστής διαλόγου θα μπορούσε να εφαρμοστεί ως μια δέσμη εντολών "If … else" , αυτή η τεχνική δεν κλιμακώνεται καλά στην πράξη, επειδή είναι δύσκολο να προβλέψουμε τους φυσικούς διάλογους, με αποτέλεσμα να αυξάνεται εκθετικά η πολυπλοκότητα αυτού του είδους μηχανής καταστάσεων. Το Botpress λύνει αυτό το πρόβλημα συνδυάζοντας διάφορες τεχνικές τεχνητής νοημοσύνης και μηχανικής μάθησης ώστε ο DM του να μπορεί να ανταπεξέλθει σε περίπλοκες συνομιλίες [5].



Σχήμα 5.2 Κύκλος ζωής ενός chatbot στο BotPress [5]

Σημείωση: τα κόκκινα σημεία αποτελούν κομμάτια στα οποία μπορεί να παρέμβει ένας προγραμματιστής και να τα τροποποιήσει ανάλογα με τις ανάγκες του chatbot που θέλει να δημιουργήσει.

5.3.2 Dialog Engine

Το Botpress χρησιμοποιεί Dialog Engine για να χειρίζεται τις διάφορες συνομιλίες. Το Dialog Engine είναι υπεύθυνο για κάθε αλληλεπίδραση ενός χρήστη με ένα chatbot, αφού διαχειρίζεται την είσοδο του χρήστη και την απόκριση του bot. Άρα, περιλαμβάνει την επεξεργασία (processing) και την απόφαση (Dialog Manager) που αναφέρθηκαν πιο πάνω.

Το Dialiog Engine χρησιμοποιεί Ροές (Flows) που αντιπροσωπεύουν τη συνολική λογική συνομιλίας ενός bot. Μια ροή αποτελείται από κόμβους (Nodes) που εκτελούν μια σειρά εντολών. Οι εντολές αποτελούν μέρος ενός κύκλου ζωής

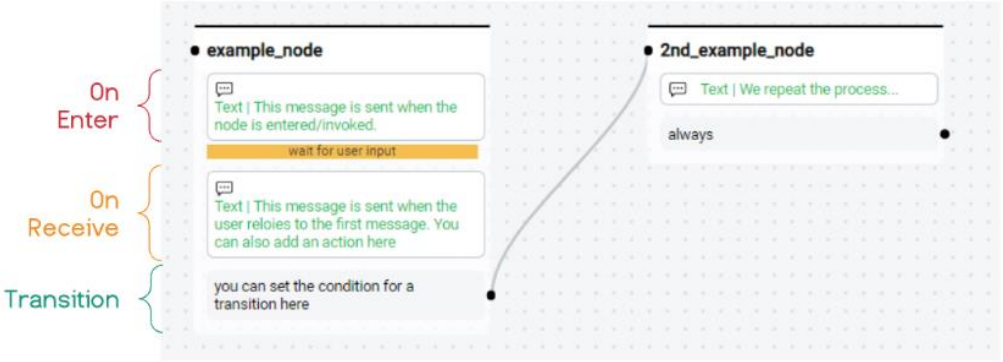
κόμβου και μπορούν να εκτελέσουν Ενέργειες (Actions). Μια ενέργεια είναι σειρά εντολών, πρόγραμμα, σε JavaScript, που είτε γράφτηκε από τον δημιουργό του chatbot είτε παρέχεται από το Botpress.

Αρχικά, τα σύνθετα bot αναλύονται σε πολλαπλές μικρότερες ροές αντί για μια, μεγάλη ροή. Ο λόγος για τη διάσπαση του bot σε πολλαπλές ροές είναι η διευκόλυνση της δυνατότητας συντήρησης και επαναχρησιμοποίησης του chatbot [5].

Πιο κάτω περιγράφονται συνοπτικά οι κύκλοι ζωής των ροών και των κόμβων:

- **Κύκλος ζωής ροής:** Μια ροή ξεκινά πάντα από τον startNode του αρχείου *.flow.json. Ο κόμβος έναρξης (startNode) υποδεικνύει το όνομα του κόμβου από τον οποίο θα ξεκινήσει η κάθε ροή. Μόλις επιλεγεί ο κόμβος, το Dialog Engine θα βάλει στη σειρά τις οδηγίες του ενεργού κόμβου (περισσότερες λεπτομερείς για τους κόμβους δίνονται πιο κάτω) και στη συνέχεια θα επεξεργαστεί τις οδηγίες διαδοχικά. Το Dialog Engine βασίζεται σε συμβάντα, πράγμα που σημαίνει ότι μια ροή θα εκτελέσει ό,τι μπορεί μέχρι να χρειαστεί να περιμένει. Μια ροή περιμένει αν ένας κόμβος επισημαίνεται ως αναμονή για κάποια είσοδο από τον χρήστη ή αν ένας κόμβος δεν μπορούσε να ταιριάζει με μια συνθήκη για μετάβαση σε άλλο κόμβο. Μόλις ολοκληρωθεί η επεξεργασία του πρώτου κόμβου, το Dialog Engine θα προχωρήσει στον επόμενο κόμβο της ροής μέχρι να φτάσει στο τέλος [5].

- **Κύκλος ζωής κόμβων:** Οι κόμβοι είναι οι κύριες μονάδες της λογικής συνομιλίας ενός bot. Μια ενεργή συνομιλία (η οποία ονομάζεται "συνεδρία") έχει πάντα έναν και μόνο ενεργό κόμβο. Ένας κόμβος μεταβαίνει σε έναν άλλο κόμβο ή ροή. Όταν δεν το κάνει, η συζήτηση έχει τελειώσει, πράγμα που σημαίνει ότι το επόμενο μήνυμα από τον χρήστη θα είναι μέρος μιας εντελώς νέας συνεδρίας. Όλοι οι κόμβοι μιας ροής χωρίζονται σε τρία μέρη, το onEnter, onReceive και Transition. Στο onEnter κομμάτι κάθε κόμβου ορίζονται οι ενέργειες που θα εκτελεστούν όταν εισέλθουμε στον κόμβο. Αν έχουμε πολλές ενέργειες τότε αυτές εκτελούνται διαδοχικά. Στο onReceive κομμάτι ορίζονται οι εντολές που θα εκτελεστούν όταν ο κόμβος λάβει ένα μήνυμα ενώ βρίσκεται σε ενεργή κατάσταση, δηλαδή δεν έχουμε μετακινηθεί από αυτόν. Το onReceive κομμάτι ενός κόμβου μπορεί να μην είναι ενεργοποιημένο ή και να περιέχει απλά εντολή αναμονής που αναγκάζει την ροή να περιμένει input από τον χρήστη. Τέλος, στο Transition κομμάτι ενός κόμβου τοποθετούνται οι συνθήκες αξιολόγησης για μετάβαση σε επόμενο κόμβο ή άλλη ροή. Κάθε Transition έχει ένα στόχο, προορισμό, στον οποίο μεταφέρει τη ροή συζήτησης όταν το Transition γίνεται αληθές. Αυτός μπορεί να είναι είτε μια άλλη ροή είτε ένας άλλος κόμβος, είτε μια προηγούμενη ροή, είτε το τέλος της συνομιλίας, είτε ακόμη και ο ίδιος (επαναφορά στο onEnter του εαυτού του). Υπάρχουν και ειδικοί κόμβοι, οι οποίοι ονομάζονται skills. Τα skill nodes χωρίζονται και αυτά σε onEnter, onReceive και Transition, το μόνο που διαφέρει είναι ότι στο onEnter υπάρχει υλοποιημένη μια συγκεκριμένη ικανότητα όπως να στέλνει email, να καλεί κάποιο API, να γεμίζει slots ή να εμφανίζει κάποια επιλογή στον χρήστη (dropdown list ή multiple choice) [5].



Σχήμα 5.3 Κύκλος ζωής ενός Node (Κόμβου) στο BotPress [5]

Σχήμα 5.4 Κύκλος ζωής ενός Node (Κόμβου) σε JSON μορφή

Σχήμα 5.5 Παράδειγμα Flow (Ροής) στο BotPress [5]

5.3.3 Actions

Οι ενέργειες είναι λειτουργίες διακομιστή, γραμμένες σε JavaScript που εκτελούνται από το bot ως μέρος μιας ροής συνομιλίας. Μια ενέργεια εκτελείται σε ένα Virtual Machine Node.js ώστε να αποτραπεί η συντριβή του διακομιστή σε περίπτωση σφάλματος. Οι ενέργειες έχουν τη δυνατότητα να αλλάξουν την κατάσταση της συνομιλίας, να στείλουν προσαρμοσμένα μηνύματα στη συνομιλία, να εκτελέσουν αυθαίρετο κώδικα, όπως κλήση ενός API ή αποθήκευση δεδομένων στη βάση δεδομένων και πολλά άλλα.

Οι ενέργειες καλούνται από εντολές που βρίσκονται στα κομμάτια onEnter και onReceive ενός κόμβου. Όταν μια ενέργεια καλείται από τον Διαχειριστή διαλόγου (DM), λαμβάνει τα ακόλουθα ορίσματα:

- **User:** Περιλαμβάνει όλα τα χαρακτηριστικά ενός χρήστη.
- **Session:** Περιλαμβάνει μεταβλητές που διατηρούνται για τη διάρκεια της συνεδρίας.
- **Temp:** Περιέχει μεταβλητές που είναι ζωντανές μόνο για τη διάρκεια της ροής.
- **Bot:** Αντικείμενο που περιέχει καθολικές μεταβλητές για το συγκεκριμένο bot (το ίδιο για όλους τους χρήστες).
- **Event:** Το αρχικό (πιο πρόσφατο) συμβάν που λήφθηκε από τον χρήστη στη συνομιλία.
- **Args:** Τα ορίσματα που χρειάζεται να μεταβιβαστούν σε αυτήν την ενέργεια από το Visual Flow Builder και ορίζονται από τον κατασκευαστή της κάθε ενέργειας.
- **Process:** Sandboxed virtual machine που περιέχει μερικές από τις μεταβλητές env (ξεκινώντας με EXPOSED_).

Υπάρχουν ενέργειες υλοποιημένες από τους κατασκευαστές του BotPress που μπορούν να χρησιμοποιηθούν σε οποιαδήποτε ροή. Μπορούν, όμως, να κατασκευαστούν και custom ενέργειες από το δημιουργό ενός bot με σκοπό να παρέχονται επιπλέον δυνατότητες στο bot. Για να μπορούν να είναι πλήρως κατανοητές όλες οι ενέργειες πρέπει να χρησιμοποιούν σχόλια JavaDoc για να εμφανίζονται σημαντικές πληροφορίες (όνομα, περιγραφή, ορίσματα, προεπιλεγμένες τιμές) στο πρόγραμμα επεξεργασίας ροής διαλόγου [5].

```
function action(bp: typeof sdk, event: sdk.IO.IncomingEvent, args: any, { user, temp, session } = event.state) {
  /** Your code starts below */


  /**
   * Small description of your action
   * @title The title displayed in the flow editor
   * @category Custom
   * @author Your_Name
   * @param {string} name - An example string variable
   * @param {any} value - Another Example value
   */
  const myAction = async (name, value) => {


  }

  return myAction(args.name, args.value)

  /** Your code ends here */
}
```

Σχήμα 5.6 Παράδειγμα Action (ενέργειας) στο BotPress

5.3.4 NLU Engine

Το BotPress NLU engine επεξεργάζεται όλα τα εισερχόμενα μηνύματα και εκτελεί Intent Classification (Ταξινόμηση πρόθεσης), Language Identification (Αναγνώριση γλώσσας), Entity Extraction (Εξαγωγή οντοτήτων) και Slot Tagging (Προσθήκη ετικετών). Τα δεδομένα δομής που παρέχουν αυτές οι εργασίες προστίθενται στα μεταδεδομένα του μηνύματος (κάτω από το event.nlu), ώστε να είναι προσβάσιμα από τις άλλες μονάδες.

- **Intent Classification:** Το intent classification βοηθά να εντοπιστεί η πρόθεση των χρηστών. Είναι ένας ακριβής τρόπος για να κατανοήσει το bot τι προσπαθεί να πει ο χρήστης με τη χρήση λέξεων-κλειδιών. Ο τρόπος που δηλώνονται τα intents αναλύθηκε στο κεφάλαιο 3 στο υποκεφάλαιο 3.2.2 (Λογισμικό Ανάπτυξης). Ο κατασκευαστής του chatbot να μπορεί να προσθέσει ενέργειες και μεταβάσεις ροής, αναζητώντας τη μεταβλητή event.nlu.intent.name η οποία κρατά το intent που αναγνωρίστηκε από το NLU engine.

Σχήμα 5.7 Παράδειγμα Transition όταν αναγνωριστεί intent

- **Language Identification:** Αναγνωρίζει τη γλώσσα στην οποία γράφει ο χρήστης, ώστε το chatbot να απαντά στην ίδια γλώσσα.

- **Entity Extraction:** Η εξαγωγή οντοτήτων βοηθά στην εξαγωγή και κανονικοποίηση γνωστών οντοτήτων από φράσεις. Ο τρόπος που δημιουργούμε entities αναλύθηκε στο κεφάλαιο 3 στο υποκεφάλαιο 3.2.2(Λογισμικό Ανάπτυξης).

- **Slot Tagging:** Το NLU engine επισημαίνει κάθε λέξη (token) που δίνει ο χρήστης σαν είσοδο. Η διαδικασία αυτή ονομάζεται slot tagging. Κάθε εντοπισμένο slot είναι προσβάσιμο στον χάρτη event.nlu.slots χρησιμοποιώντας το όνομά του ως κλειδί. Τα slots συχνά χρησιμοποιούνται σε συνδυασμό με τα entities και intents με σκοπό να γίνεται tag ένα entity σε ένα intent. Δηλαδή, μπορούν να χρησιμοποιηθούν για να αναγνωριστεί μια λέξη κλειδί σε μια πρόταση. Για παράδειγμα, αν ο χρήστης ζητά **drug** allergies ή **food** allergies και όχι απλά allergies, το drug και το food είναι δυο διαφορετικά slots και υποδεικνύουν ότι ο χρήστης ζητά συγκεκριμένη πληροφορία [5].

5.4 Εγκατάσταση και τρόπος λειτουργίας BotPress server

Ο τρόπος εγκατάστασης και λειτουργίας του server βρίσκεται στο Παράρτημα Α: Εγκατάσταση BotPress server.

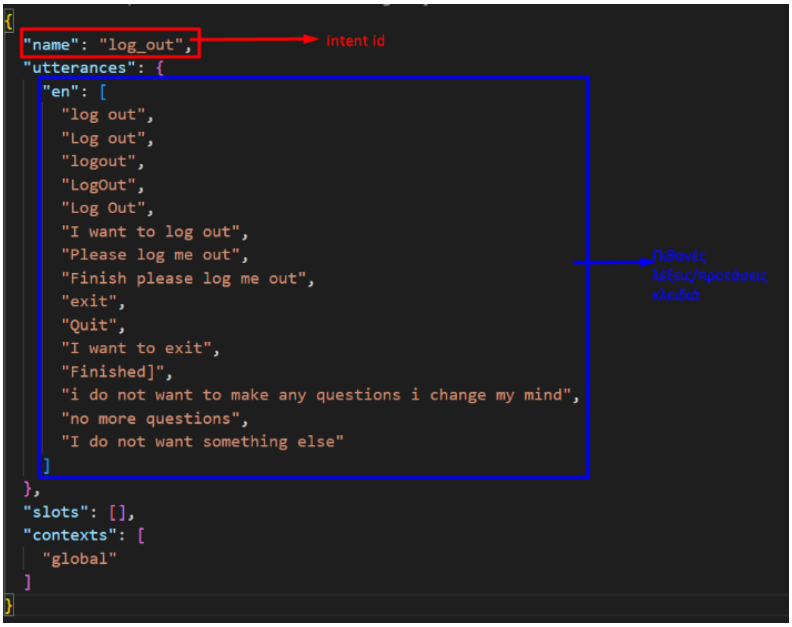
5.5 Σχεδιασμός και υλοποίηση του chatbot

Το chatbot για την παρούσα ατομική διπλωματική εργασία αναπτύχθηκε στην πλατφόρμα BotPress σε συνδυασμό με MySQL και Knex για την άντληση δεδομένων από τη βάση δεδομένων και HTML και CSS για τη δημιουργία της ιστοσελίδας που αναπτύχθηκε για σκοπούς δοκιμής του chatbot. Στο κεφάλαιο αυτό παρουσιάζεται συνοπτικά η σχεδίαση του παρόντος συστήματος και ο τρόπος που υλοποιήθηκε με τη χρήση των τεχνολογιών που προαναφέρθηκαν, καθώς επίσης και οθόνες με παραδείγματα.

5.5.1 Intents – Entities

Τα intents και τα entities αποτελούν βασικό κομμάτι της υλοποίησης του chatbot, αφού είναι τα κύρια συστατικά μέρη του NLU engine μέσω του οποίου κατανοείται η είσοδος του χρήστη.

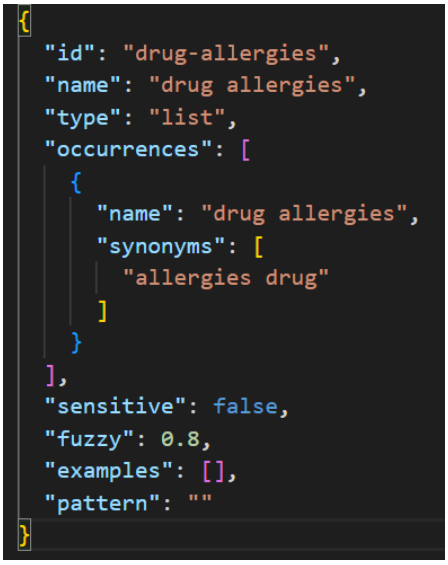
Αρχικά, κάθε intent αποτελεί ένα *.json αρχείο το οποίο περιέχει τη δομή του. Στο παρόν chatbot έχουν δημιουργηθεί εννιά intents, ένα για κάθε κατηγορία ερωτήσεων που μπορεί να απευθύνει ο χρήστης στο chatbot,ένα για βοήθεια και ένα για έξοδο από το σύστημα. Τα ids των intents είναι “allergies”, “current_problems”, “medical_devices”, “past_illnesses”, “prescription_summary”, “surgeries”, “vaccinations”, “help” και “log_out”.



Σχήμα 5.8 Παράδειγμα intent: log_out.json

Όσον αφορά τα entities που δημιουργήθηκαν, σκοπό έχουν την απομόνωση λέξεων κλειδιών από ερωτήσεις που απευθύνει ο χρήστης. Παραδείγματος χάριν, όταν ο χρήστης ζητά τα drug allergies του και όχι απλά allergies, αναγνωρίζεται η λέξη drug, η οποία προσδιορίζει συγκεκριμένο χαρακτηριστικό αυτού που ψάχνει ο χρήστης. Στο παρόν σύστημα, η δυνατότητα αναγνώρισης χαρακτηριστικής λέξης κλειδί δίνεται μόνο στις περιπτώσεις που ο χρήστης ζητά να δει αλλεργίες. Για τον σκοπό αυτό δημιουργήθηκαν έξι entities και έξι slots, ένα για κάθε διαφορετικό τύπο αλλεργιών που είναι πιθανόν να ζητήσει ο χρήστης.

Κάθε entity αποτελεί και αυτό ένα *.json αρχείο με την πιο κάτω δομή:

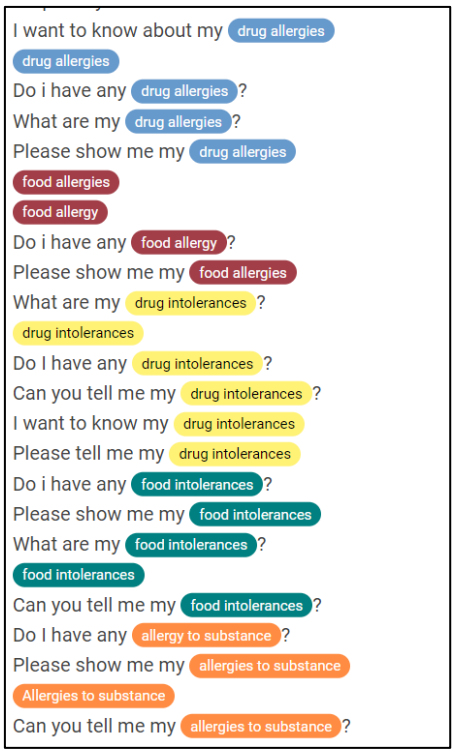


Σχήμα 5.9 Παράδειγμα entity: drug-allergies.json

Αφού δημιουργήθηκαν τα entities, δημιουργήθηκαν και έξι slots. Τα slots δημιουργούνται μέσα στο αρχείο .json του αντίστοιχου intent. Στην περίπτωση αυτή, τα slots δημιουργήθηκαν στο αρχείο allergies.json στο folder intents. Κάθε ένα από τα slots φέρουν το όνομα του entity στο οποίο αναφέρονται.

Σχήμα 5.10 Παράδειγμα με slots

Για να γίνουν πιο κατανοητά τα slots που δημιουργήθηκαν, πιο κάτω φαίνονται τα slots στο BotPress studio. Επίσης, φαίνονται με χαρακτηριστικά χρώματα μέσα στις προτάσεις, λέξεις κλειδιά του intent allergies.



Σχήμα 5.11 Παράδειγμα με slots στο BotPress Studio

5.5.2 Databases

Το chatbot αντλεί δεδομένα μέσω δύο βάσεων δεδομένων MySQL. Οι δύο αυτές βάσεις δημιουργήθηκαν στα πλαίσια αυτής της Διπλωματικής εργασίας, ώστε να είναι δυνατή η εισαγωγή δεδομένων για δοκιμή (test data).

Οι βάσεις είναι:

1. **patients_info:** Η βάση αυτή περιέχει τα προσωπικά στοιχεία των ασθενών. Δηλαδή, όνομα, επίθετο, ημερομηνία γέννησης, ταυτότητα και τα λοιπά. Πιο συγκεκριμένα, περιέχει ένα πίνακα personal_info στον οποίο αποθηκεύονται όλες αυτές οι πληροφορίες. Η βάση αυτή χρησιμοποιείται για να γίνεται ταυτοποίηση του χρήστη. Όταν ο χρήστης δώσει στο chatbot την ταυτότητά του, τότε αυτό ψάχνει στη συγκεκριμένη βάση να βρει την αντίστοιχη ταυτότητα και καλωσορίζει τον χρήστη με το όνομά του, δηλαδή, το όνομα που αντιστοιχεί στην ταυτότητα που εντόπισε. Όταν ταυτοποιηθεί ο χρήστης, το chatbot, εντοπίζει την ημερομηνία γέννησης που, σε συνδυασμό με την ταυτότητα, χρησιμοποιούνται στην εύρεση του patient_id. Το patient_id του χρήστη εξασφαλίζει πρόσβαση στα ιατρικά δεδομένα που βρίσκονται στη δεύτερη βάση δεδομένων. Περισσότερα για τον τρόπο που συνδέονται οι δύο βάσεις αναλύονται στη συνέχεια του κεφαλαίου.

2. **myDatabase:** Η βάση αυτή περιέχει όλα τα ιατρικά δεδομένα του patient summary και e-Prescription όλων των ασθενών. Δημιουργήθηκε με τη χρήση

script, ώστε να είναι ένα ακριβές αντίγραφο της βάσης δεδομένων που χρησιμοποιεί το ήδη υλοποιημένο σύστημα Endorse. Ο κάθε πίνακας της βάσης κρατά δεδομένα που αφορούν ένα συγκεκριμένο κομμάτι του patient summary και e-Prescription. Για παράδειγμα, ο πίνακας allergies_mapping κρατά όλες τις αλλεργίες του κάθε ασθενή. Κύριο κλειδί σε όλους τους πίνακες αποτελεί το patient_id, το οποίο, όπως αναφέρθηκε πιο πάνω και εξηγείται και στη συνέχεια, διαφέρει από την πραγματική ταυτότητα του χρήστη. Επίσης, στη βάση περιέχονται οι πίνακες patient_data, codings και codesystems, οι οποίοι αποτελούν βασικό κομμάτι στην εύρεση των ιατρικών δεδομένων. Στο patient_data υπάρχουν αποθηκευμένα το patient_id του κάθε ασθενή και το αντίστοιχο patient_hash, το οποίο παράγεται βάσει της πραγματικής ταυτότητας του χρήστη και της ημερομηνίας γέννησής του. Στο codings και στο codesystems κρατούνται όλοι οι κωδικοί και οι αντίστοιχες περιγραφές τους. Όλα τα ιατρικά δεδομένα είναι καταχωρημένα στους διάφορους πίνακες με coding ids τα οποία αντιστοιχούν σε μια λεκτική περιγραφή στους δύο προαναφερθέντες πίνακες. Οι κωδικοί αυτοί δεν είναι τυχαία νούμερα, αλλά προέρχονται από το MVC (Master Value Catalog).

Οι δύο αυτές βάσεις, όπως προαναφέρθηκε, συνδέονται μέσω του patient_hash. Το patient_hash δημιουργείται μέσω της ταυτότητας και της ημερομηνίας γέννησης του κάθε χρήστη, δεδομένα τα οποία βρίσκουμε μέσω της βάσης patients_info. Αρχικά, δημιουργείται ένα string, το οποίο είναι η ταυτότητα ενωμένη με την ημερομηνία γέννησης, για παράδειγμα “101010101999-03-02”. Το string αυτό κρυπτογραφείται με τη χρήση του αλγόριθμου κρυπτογράφησης AES. Το αποτέλεσμα που θα παραχθεί από το AES encryption γίνεται hash με τη χρήση του αλγορίθμου md5. Το τελικό αποτέλεσμα καταχωρείται στη βάση myDatabase στον πίνακα patient_data, μαζί με ένα αριθμό ο οποίος αποτελεί το patient_id του χρήστη.

Για να επιτευχθεί η σύνδεση του chatbot στις βάσεις δεδομένων, καθώς επίσης και η εύρεση του patient_hash κάθε ασθενή, δημιουργήθηκαν διάφορα custom actions με τη χρήστη του Knex , τα οποία επεξηγούνται και αναλύονται σε πιο κάτω υποκεφάλαιο.

5.5.3 Ροές (Flows)

Αρχικά το παρών chatbot αποτελείται από έντεκα (11) ροές (Flows). Κάθε flow δημιουργείται μέσω δύο αρχείων json, το *.flow.json και το *.ui.json. Το αρχείο *.flow.json περιλαμβάνει ουσιαστικά το flow σαν δομή και μέσα σε αυτό δηλώνονται τα nodes της ροής αυτής. Στο *.ui.json, περιέχονται στοιχεία για την αναπαράσταση του flow στο BotPress Studio, ώστε να είναι πιο εύκολη η κατανόηση και αποσφαλμάτωση του.

Οι ροές που δημιουργήθηκαν για το παρών chatbot είναι οι ακόλουθες:

1. **Main**: Το flow αυτό αποτελεί το κεντρικό μονοπάτι συζήτησης και από αυτό ξεκινά κάθε συνεδρία με το chatbot. Αποτελείται από δεκατέσσερα nodes, εκ των οποίων τα δύο αποτελούν skills και συγκεκριμένα choice node. Μέσω της ροής αυτής, γίνεται η ταυτοποίηση του χρήστη ώστε να μπορέσει να ξεκινήσει μια συνεδρία. Περισσότερα για την ταυτοποίηση του χρήστη ακολουθούν πιο κάτω στο κεφάλαιο αυτό.

2. **Error handling**: Το flow αυτό διαχειρίζεται τα σφάλματα ανά πάσα χρονική στιγμή. Σε περίπτωση, δηλαδή, που εμφανιστεί κάποιο μη προγραμματισμένο σφάλμα η

συνεδρία μεταφέρεται σε αυτή τη ροή. Με τον όρο μη προγραμματισμένο σφάλμα εννοούμε ένα σφάλμα το οποίο δεν προβλέψαμε ότι μπορεί να συμβεί ή υπό κανονικές συνθήκες δεν πρέπει να συμβαίνει, όπως για παράδειγμα να μην ανταποκρίνεται η βάση δεδομένων. Σε αυτές τις περιπτώσεις, το chatbot μεταφέρεται σε αυτή την ροή και διαχειρίζεται το σφάλμα. Η ροή αυτή αποτελείται από ένα μόνο κόμβο ο οποίος εμφανίζει μήνυμα στον χρήστη και σταματά τη συνεδρία.

3. **allergies**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων αλλεργίες που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τις αλλεργίες του. Η ροή αποτελείται από δεκατρείς κόμβους, εκ των οποίων ο ένας αποτελεί choice node.

4. **current_problems**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων τρέχοντα προβλήματα που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τα τρέχοντα προβλήματα που αντιμετωπίζει.

5. **end_flow**: Η ροή αυτή τερματίζει την συνεδρία του χρήστη με το chatbot. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να αποσυνδεθεί από το chatbot ή αν απαντήσει σε κάποιο skill ότι δεν επιθυμεί να υποβάλει κάποια άλλη ερώτηση.

6. **help**: Η ροή αυτή σκοπό έχει την παροχή βοήθειας στον χρήστη αν αυτός τη ζητήσει. Όταν ο χρήστης μεταφερθεί στη ροή αυτή εμφανίζονται πληροφορίες για το πώς λειτουργεί το chatbot.

7. **medical_devices_implants**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων ιατρικών συσκευών και εμφυτευμάτων που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τις ιατρικές συσκευές και τα εμφυτεύματά του.

8. **past_illnesses**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων παλιές ασθένειες που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τις ασθένειες που είναι καταχωρημένες στο ιατρικό ιστορικό του.

9. **prescription_summary**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων περίληψη φαρμακευτικής αγωγής που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για το ιστορικό φαρμακευτικής αγωγής του.

10. **surgeries**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων επεμβάσεις που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τις επεμβάσεις που έχει υποβληθεί.

11. **vaccinations**: Η ροή αυτή αντιπροσωπεύει την κατηγορία ερωτήσεων εμβολιασμοί που παρουσιάζεται στο κεφάλαιο 4. Ο χρήστης μεταφέρεται στη ροή αυτή αν ζητήσει να μάθει πληροφορίες για τους εμβολιασμούς του.

Σημείωση: Ο κώδικας του κάθε flow παραδόθηκε στο αρχείο .zip και το *.flow.json αρχείο παρουσιάζεται στο Παράρτημα Β: Flows αυτού του δοκιμίου.

Στην κύρια ροή (Main) κομβικά κομμάτια αποτελούν η **ταυτοποίηση** του χρήστη και η **αναγνώριση της ερώτησης** που υποβάλλει ο χρήστης.

- **Ταυτοποίηση:** Η ταυτοποίηση του χρήστη γίνεται στον κόμβο log-in στο Main flow. Αρχικά, στο onEnter του κόμβου το chatbot ζητά από τον χρήστη να πληκτρολογήσει την ταυτότητά του. Στο onReceive ο κόμβος περιμένει για είσοδο του χρήστη και όταν λάβει κάποια είσοδο δημιουργεί μεταβλητή userId η οποία λαμβάνει το input του χρήστη και το κρατά σε επίπεδο session, δηλαδή για ολόκληρη τη συνεδρία. Αφού γίνει αυτό, περνούμε στο Transition κομμάτι του κόμβου στο οποίο υπάρχουν τέσσερα transitions. Πρώτα, ελέγχεται αν ο χρήστης ζήτησε βοήθεια αντί να έδωσε την ταυτότητα του (if intent is help: next 'before log-in help'). Στην περίπτωση αυτή, το chatbot μεταφέρεται σε ένα άλλο κόμβο της ίδιας ροής με το όνομα "before log-in help", ο οποίος τυπώνει στον χρήστη μήνυμα που εξηγεί τι πρέπει να κάνει ο χρήστης για να ταυτοποιηθεί. Το δεύτερο transition ελέγχει ότι η είσοδος του χρήστη είναι αριθμός καλώντας τη συνάρτηση isNaN() της JavaScript. Αν ο χρήστης δεν έδωσε αριθμό, τότε μεταφερόμαστε στο "wrong id" κόμβο, ο οποίος επιστρέφει στον χρήστη ότι η ταυτότητα που πληκτρολόγησε είναι λάθος και του ζητά να ξαναπληκτρολογήσει την ταυτότητά του. Το επόμενο transition ελέγχει ότι υπάρχει όντως είσοδος από τον χρήστη και μεταφέρει τη ροή στον κόμβο "check-for-id-in-database" ο οποίος εκτελεί το custom action "exist-check" που περιγράφεται πιο κάτω μαζί με τα υπόλοιπα actions που δημιουργήθηκαν. Τέλος, έχουμε το transition "Otherwise" το οποίο σε περίπτωση που κανένα από τα προηγούμενα δεν γίνει αληθές, μεταφέρει την ροή στον κόμβο "Misunderstood" ο οποίος τυπώνει στον χρήστη μήνυμα ότι το chatbot δεν κατανόησε τι του έδωσε σαν input και επιστρέφει τη ροή συζήτησης πίσω στον κόμβο "log-in". Η ταυτοποίηση ολοκληρώνεται μέσω του κόμβου "check-for-id-in-database" ο οποίος ελέγχει για το στοιχεία του ασθενή στη βάση. Αν ο ασθενής είναι καταχωρημένος στη βάση patients_info, ακολουθεί η εύρεση των ιατρικών δεδομένων του, τα οποία βρίσκονται σε άλλη βάση δεδομένων.

- **Αναγνώριση ερωτήσεων:** η αναγνώριση του περιεχομένου της ερώτησης που υποβάλλει ο χρήστης, γίνεται στον κόμβο questions_answers. Στο onReceive ο κόμβος περιμένει για την απάντηση του χρήστη και στο Transition κομμάτι ελέγχει αν η είσοδος που έδωσε ο χρήστης εμπίπτει σε κάποιο intent ή entity βάσει του τι αναγνώρισε το NLU engine. Για να αναγνωριστεί σε ποιο intent ή entity εμπίπτει η ερώτηση, ελέγχονται οι τιμές των μεταβλητών event.nlu.intent.name και event.nlu.slots με το αντίστοιχο όνομα. Ανάλογα με αυτές τις τιμές, μεταφερόμαστε και στην αντίστοιχη επόμενη ροή. Όλες οι ροές επιστρέφουν όταν ολοκληρωθούν πίσω στον κόμβο questions_answers, ώστε το chatbot να αναγνωρίσει την επόμενη ερώτηση. Πιο κάτω απεικονίζονται τα transitions που περιέχονται στον κόμβο αυτό:

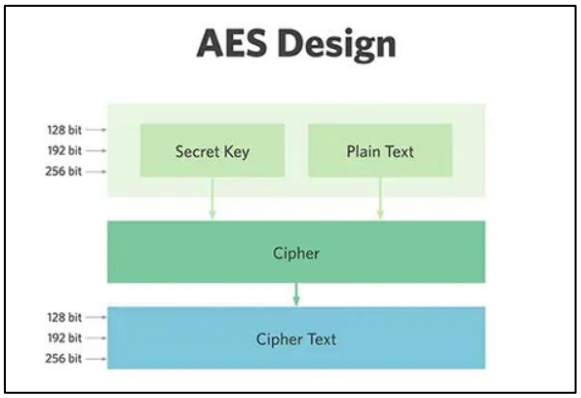
Σχήμα 5.12 Τα Transitions του questions_answers node

5.5.4 Custom Actions

Τα Custom Actions, είναι ενέργειες που δημιουργεί ο κατασκευαστής του chatbot με σκοπό να προσθέσει κάποια επιπλέον ικανότητα στο bot. Είναι και αυτά, όπως όλες οι ενέργειες, προγράμματα γραμμένα σε JavaScript.

Για την υλοποίηση του παρόντος συστήματος δημιουργήθηκαν τα ακόλουθα custom actions:

- **AES_encryption.js** : Το action αυτό παίρνει σαν παραμέτρους id και date, δηλαδή την (πραγματική) ταυτότητα του χρήστη και την ημερομηνία γέννησής του, δημιουργεί string και ακολούθως το κρυπτογραφεί βάση του αλγόριθμου κρυπτογράφησης AES δεδομένου του δοσμένου secret key και επιστρέφει το αποτέλεσμα.



Σχήμα 5.13 AES encryption algorithm

Η γενική ιδέα του AES αλγόριθμου είναι αυτή που απεικονίζεται στο πιο πάνω σχήμα. Στο cipher κομμάτι ο αλγόριθμος AES περιλαμβάνει έναν γύρο που ονομάζεται AddRoundKey, δέκα γύρους που περιλαμβάνουν τέσσερις βασικές πράξεις- SubBytes, ShiftRows, MixColums, AddRoundKey- και τον τελικό γύρο που περιλαμβάνει μόνο τις πράξεις SubBytes, ShiftRows, AddRoundKey. Η JavaScript παρέχει έτοιμες συναρτήσεις που απλοποιούν τη διαδικασία υλοποίησης του αλγόριθμου AES.

- **exist-check.js** : Το action αυτό παίρνει σαν παραμέτρους, table, key, whereColumn, value, database και hash. Σκοπός του action αυτού είναι να ενώνεται στο περιβάλλον των δύο βάσεων και να ελέγχει αν υπάρχει το key μέσα στον πίνακα table της βάσης δεδομένων database, όπου η τιμή της στήλης whereColumn ισούται με το value, και να το επιστρέφει. Η παράμετρος hash είναι παράμετρος 0 ή 1, ώστε να γνωρίζουμε αν ψάχνουμε να βρούμε hash value στη βάση. Ο λόγος ύπαρξης αυτής της παραμέτρου, είναι το ότι η html από μόνη της αντικαθιστά χαρακτήρες όταν η τιμή μεταφέρεται μέσω πράξεων. Άρα, όταν περνούμε το hash που δημιουργήσαμε με το AES, χρειάζεται να επεξεργαστεί το value πριν ψάξουμε να βρούμε το id με το αντίστοιχο hash. Επίσης, χρειάζεται στο query που εκτελούμε στη βάση να καλέσουμε τον αλγόριθμο md5 για να βρούμε το σωστό id.

- **get_allergies_info.js** : Το action αυτό παίρνει σαν παραμέτρους name,id και category. Σκοπός του είναι να ενώνεται στη βάση με τα ιατρικά δεδομένα, να

βρίσκει και να επιστρέφει τις πληροφορίες που περιέχονται στον πίνακα allergies_mapping. Οι αλλεργίες, όπως αναφέρθηκε ήδη, χωρίζονται σε πολλές κατηγορίες. Η παράμετρος name αντιστοιχεί στο όνομα της γενικής κατηγορίας που ψάχνουμε (allergy, intolerance ή propensity), ενώ το category αναφέρεται στην ειδική κατηγορία (food allergy, drug allergy, allergy to substance κτλ). Το id είναι το patient_id που βρήκαμε δεδομένου του patient_hash.

- **get_illnesses_info.js** : Το action αυτό παίρνει σαν παραμέτρους το id και το ongoing. Το id όπως και στο προηγούμενο action είναι η ταυτότητα στη βάση με τα ιατρικά δεδομένα, δηλαδή το patient_id. Το ongoing είναι μια παράμετρος που μπορεί να είναι 0 ή 1 ή 6. Το 0 αντιστοιχεί σε ασθένειες από τις οποίες ο χρήστης έχει αναρρώσει. Το 1 αντιστοιχεί σε ασθένειες από τις οποίες ο χρήστης συνεχίζει να νοσεί και τέλος το 6 αντιστοιχεί σε ασθένειες που πέρασε ο χρήστης τους τελευταίους 6 μήνες. Το action αυτό ενώνεται στο περιβάλλον των βάσεων και βρίσκει τα δεδομένα από τη βάση με τα ιατρικά δεδομένα και συγκεκριμένα από τον πίνακα illnesses_mapping.

- **get_medical_devices_info.js** : Το action αυτό παίρνει σαν παραμέτρους το id του ασθενή και το description. Το id είναι και σε αυτή την περίπτωση το id των ιατρικών δεδομένων και όχι το πραγματικό id του χρήστη. Το description μπορεί να είναι implants ή medical devices. Ανάλογα με το description, το action βρίσκει και επιστρέφει τα δεδομένα από τον πίνακα medical_devices_mapping.

- **get_prescription.js** : Το action αυτό παίρνει σαν μόνη παράμετρο το id, το οποίο είναι το patient_id. Σκοπός του είναι να εντοπίσει και να επιστρέψει τα δεδομένα που βρίσκονται στον πίνακα prescription_mapping. Δηλαδή, εντοπίζει το prescription summary του ασθενή.

- **get_social_history.js** : Το action αυτό παίρνει σαν μόνη παράμετρο το id, το οποίο είναι το patient_id. Σκοπός του είναι να εντοπίσει και να επιστρέψει τα δεδομένα που βρίσκονται στον πίνακα social_history_mapping. Δηλαδή, εντοπίζει τα social problems του ασθενή.

- **get_surgeries_info.js** : Το action αυτό παίρνει σαν μόνη παράμετρο το id, το οποίο είναι το patient_id. Σκοπός του είναι να εντοπίσει και να επιστρέψει τα δεδομένα που βρίσκονται στον πίνακα surgical_procedures_mapping. Δηλαδή, εντοπίζει τις επεμβάσεις στις οποίες έχει υποβληθεί ο ασθενής.

- **get_vaccinations_info.js** : Το action αυτό παίρνει σαν μόνη παράμετρο το id, το οποίο είναι το patient_id. Σκοπός του είναι να εντοπίσει και να επιστρέψει τα δεδομένα που βρίσκονται στον πίνακα vaccinations_mapping. Δηλαδή, εντοπίζει τους εμβολιασμούς που έχει κάνει ο ασθενής.

Όλα τα custom actions, εκτός το AES_encryption, επιστρέφουν τα δεδομένα στο user.data, το οποίο μπορούμε να διαβάσουμε σε επίπεδο ροής καλώντας {{user.data}}. Το AES_encryption επιστρέφει το value που υπολογίζει στο user.hash το οποίο μπορούμε να διαβάσουμε με ακριβώς τον ίδιο τρόπο.

Επίσης, σημαντικό κομμάτι της υλοποίησης όλων αυτών των actions, εκτός του AES_encryption, είναι το πώς ενώνονται στο περιβάλλον με τις βάσεις δεδομένων. Για να επιτευχθεί αυτό χρησιμοποιείται η τεχνολογία knex.

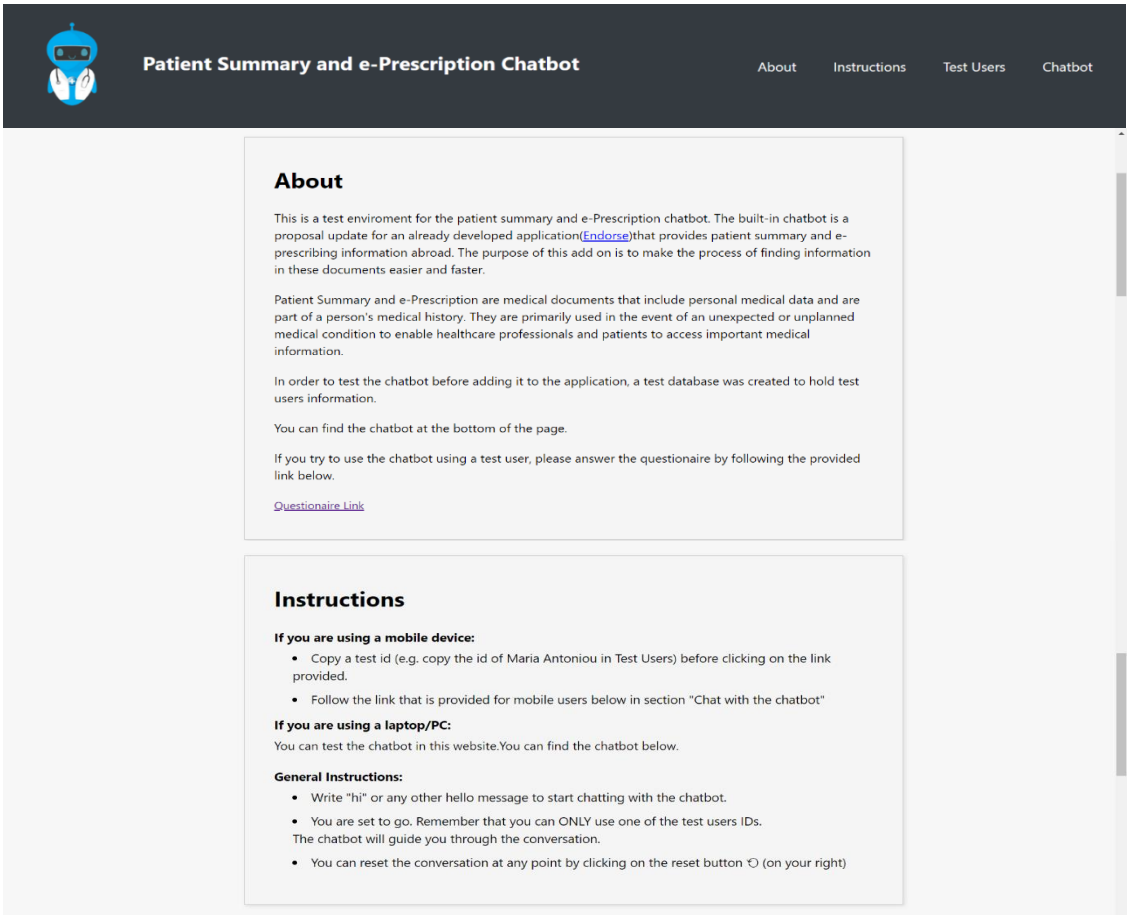
Το knex είναι ένας SQL query builder για MySQL καθώς επίσης και για πολλές άλλες βάσεις δεδομένων, ο οποίος χρησιμοποιείται κυρίως για το Node.js [13].

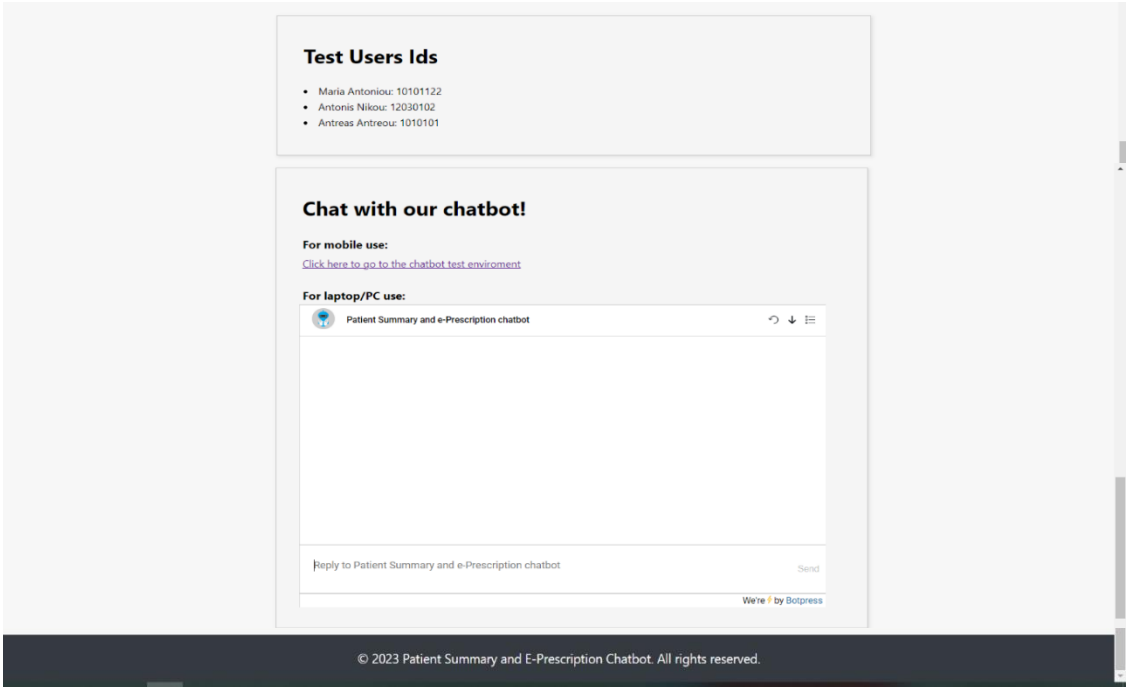
Ο κώδικας που χρησιμοποιείται για να κτιστεί το knex connection και τα queries φαίνονται στο Παράρτημα Γ: Custom Actions , όπως και όλος ο υπόλοιπος κώδικας που γράφτηκε για την υλοποίηση των actions που προαναφέρθηκαν.

Κεφάλαιο 6 – Δοκιμή και Συμπεράσματα

6.1 Σελίδα Δοκιμής

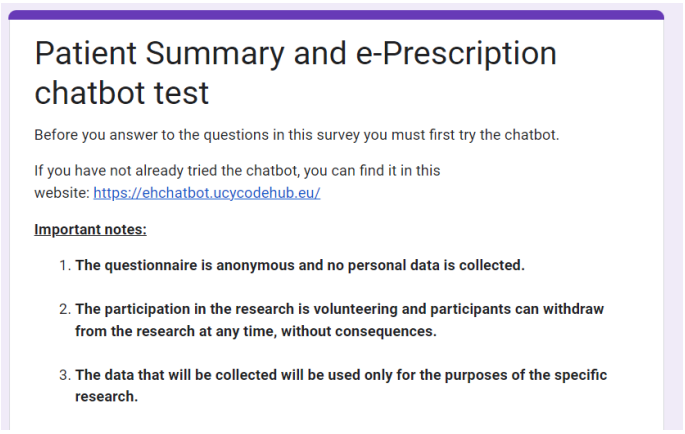
Για τη δοκιμή του chatbot που αναπτύχθηκε στα πλαίσια αυτής της Διπλωματικής Εργασίας δημιουργήθηκε μια ιστοσελίδα η οποία περιέχει σύντομη περιγραφή του συστήματος, οδηγίες χρήσης του test περιβάλλοντος του chatbot, test users και τέλος, το ίδιο το chatbot. Η σελίδα αυτή είναι πολύ απλή και αναπτύχθηκε εξ ολοκλήρου με τη χρήση html και css. Πιο κάτω παραθέτονται στιγμιότυπα της σελίδας αυτής.

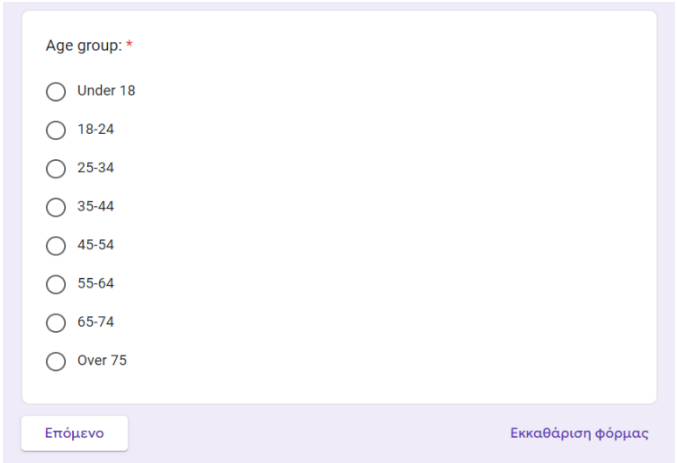




6.2 Ερωτηματολόγιο

Για την ανατροφοδότηση σε σχέση με την εμπειρία που είχαν οι χρήστες του chatbot που αναπτύχθηκε, δημιουργήθηκε ερωτηματολόγιο με τη χρήση Google Forms. Το ερωτηματολόγιο χωρίζεται σε τρία μέρη. Την εισαγωγή, στην οποία υπάρχει σύντομη περιγραφή του συστήματος και συλλέγονται στοιχεία που αφορούν μόνο την ηλικιακή ομάδα του συμμετέχοντα, τις γενικές ερωτήσεις σε σχέση με την εμπειρία του συμμετέχοντα με τη χρήση chatbot και τις ερωτήσεις που αναφέρονται στη χρήση του συστήματος της παρούσας διπλωματικής. Το ερωτηματολόγιο απεικονίζεται πιο κάτω και είναι γραμμένο στην αγγλική γλώσσα.





General Questions about the use of chatbots

How often do you use a chatbot in a website or an application to get faster information? *

- ◯ Never
- ◯ Occasionally
- ◯ Sometimes
- ◯ Always

*Answer this only if you answer occasionally, sometimes or always to the previous question

Do you prefer using chatbots to find the information you want or do you find it easier searching it on your own?

- ◯ Prefer chatbot
- ◯ Prefer search by my own

Πίσω Επόμενο Εκκαθάριση φόρμας

Feedback for the provided chatbot

Did you find the chatbot easy to use? *

	1	2	3	4	5	
really difficult	◯	◯	◯	◯	◯	really easy

Did the chatbot respond quickly to your questions? *

- ◯ Yes
- ◯ Most of the times
- ◯ No

Did the chatbot provide clear and understandable answers? *

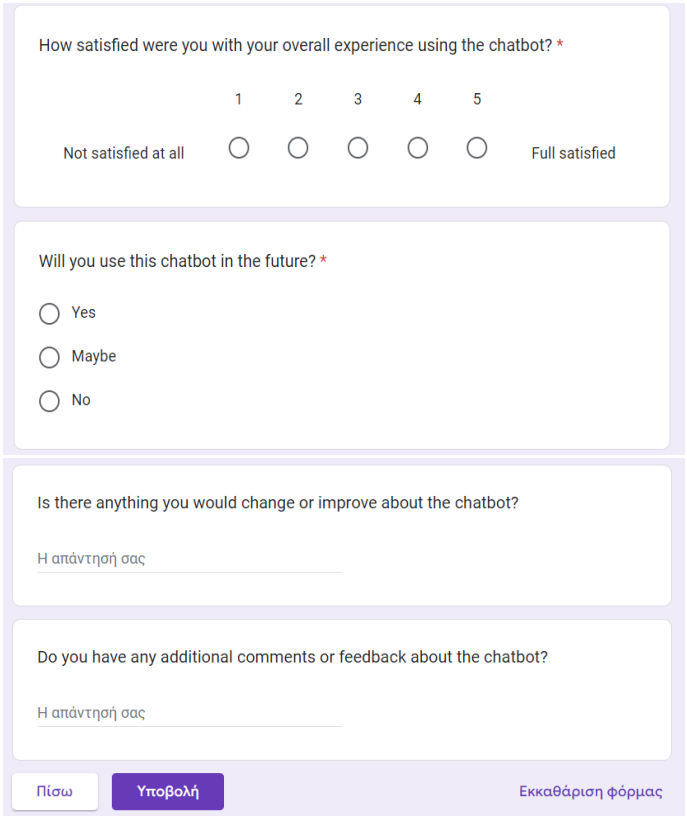
- ◯ Yes
- ◯ Most of the times
- ◯ No

Did you encounter any errors or bugs while using the chatbot? *

- ◯ Yes
- ◯ No

If yes, what were they?

Η απάντησή σας



6.3 Πλαίσιο αξιολόγησης – Συμμετέχοντες

Για την αξιολόγηση του chatbot, μέσω της σελίδας που δημιουργήθηκε, καλέστηκε μια ομάδα εθελοντών, να δοκιμάσουν το chatbot και να απαντήσουν στο σχετικό ερωτηματολόγιο. Οι συμμετέχοντες ήταν άτομα διάφορων ηλικιών και επαγγελματικών χώρων έτσι ώστε να μπορέσει να καλυφθεί όσο το δυνατό μεγαλύτερο φάσμα πιθανών χρηστών του συστήματος.

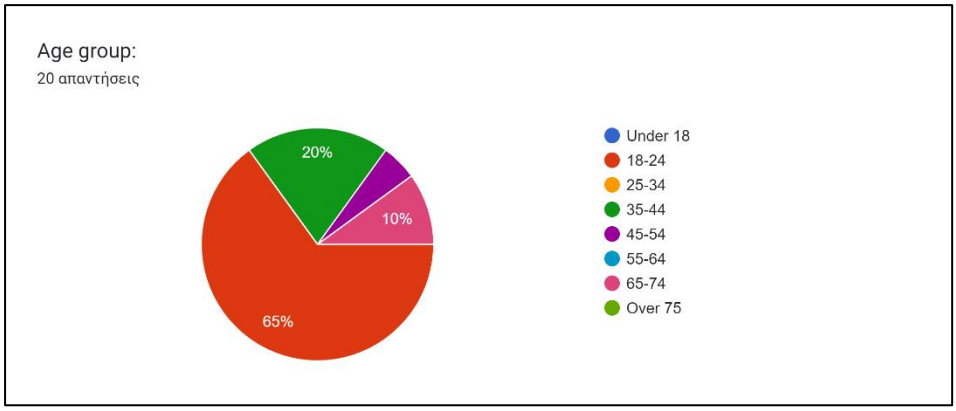
Πιο συγκεκριμένα συντάχθηκε και στάλθηκε ένα μήνυμα το οποίο εξηγεί σε συντομία τον σκοπό του chatbot και καλεί τα άτομα να δοκιμάσουν το chatbot μέσω της ιστοσελίδας δοκιμής και ακολούθως να απαντήσουν το σχετικό ερωτηματολόγιο.

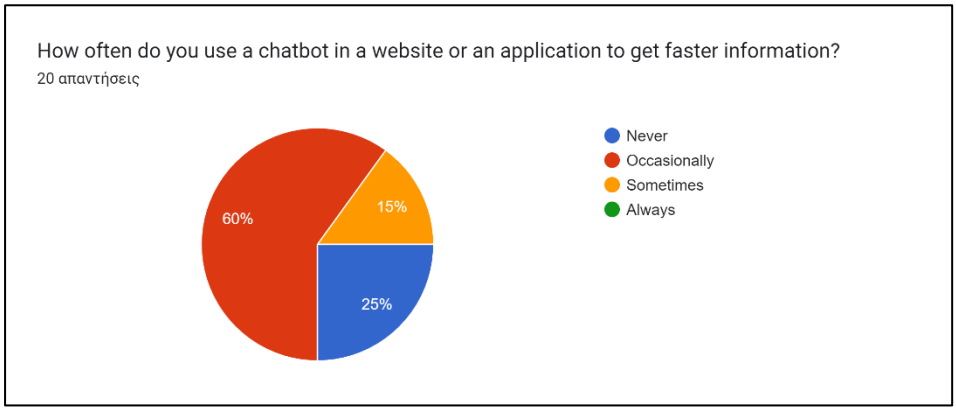
6.4 Αποτελέσματα Δοκιμής

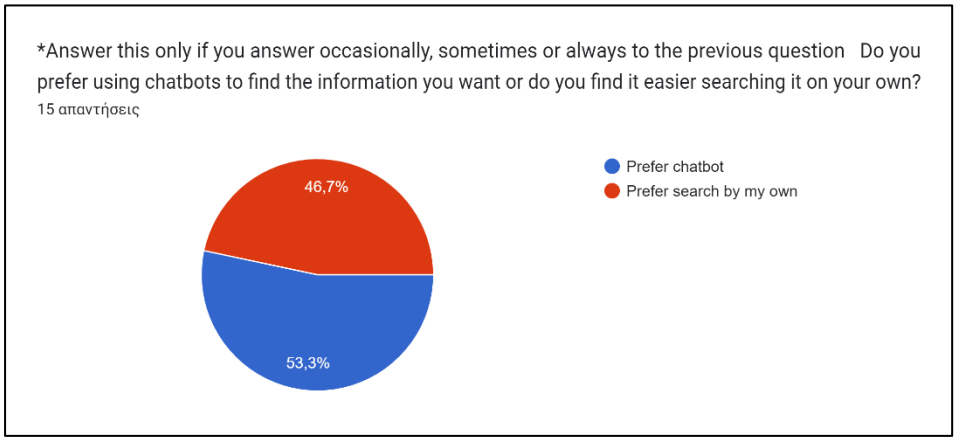
Η δοκιμή του chatbot χωρίστηκε σε δύο φάσεις. Η πρώτη φάση σκοπό είχε την επίλυση τυχών σφαλμάτων, όπως λάθος απαντήσεις του chatbot σε ερωτήσεις, ορθογραφικά λάθη και τα λοιπά, καθώς επίσης, και στο να ελεγχθεί αν θα παρουσιάζονταν οποιαδήποτε προβλήματα στη χρήση. Στη φάση αυτή, καλέστηκαν μόνο είκοσι άτομα να χρησιμοποιήσουν το σύστημα. Η δεύτερη φάση σκοπό τη συλλογή όσο το δυνατόν περισσότερων απαντήσεων, ώστε να υπάρξει μια πιο ολοκληρωμένη εικόνα σε σχέση με το αν πραγματικά το chatbot θα αποτελούσε μια καλή μελλοντική προσθήκη σε ήδη υπάρχοντα συστήματα.

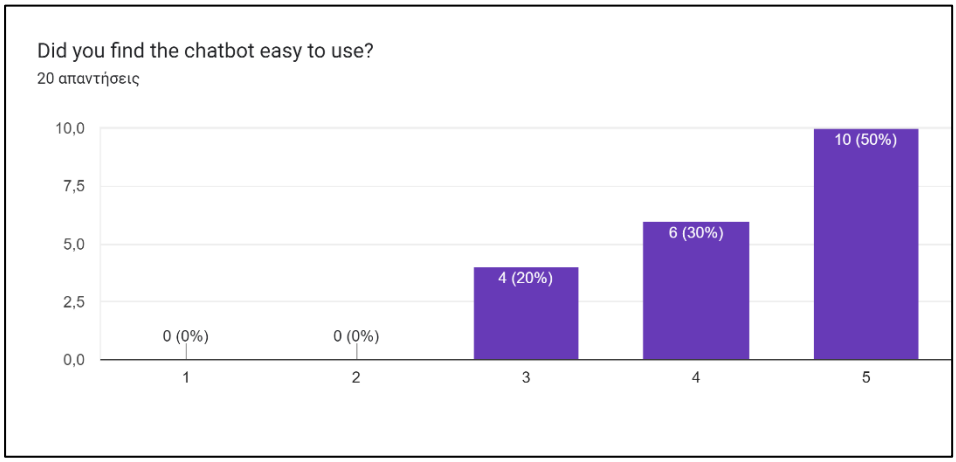
6.4.1 Αποτελέσματα Πρώτης Φάσης

Τα αποτελέσματα που συλλέχθηκαν από την πρώτη φάση της δοκιμής παρουσιάζονται πιο κάτω:







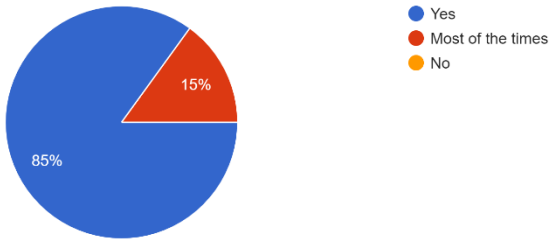


Did the chatbot respond quickly to your questions?

20 απαντήσεις

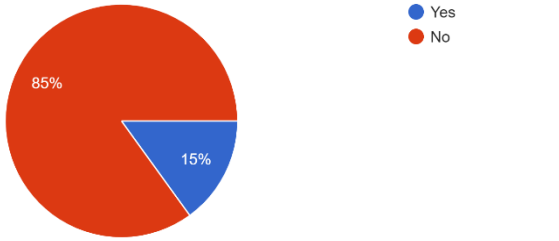
Did the chatbot provide clear and understandable answers?

20 απαντήσεις



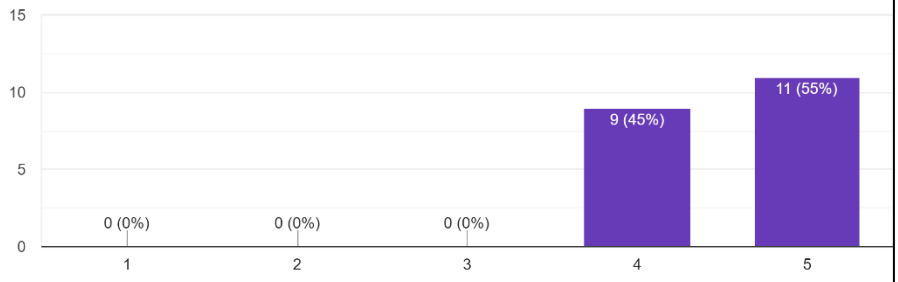
Did you encounter any errors or bugs while using the chatbot?

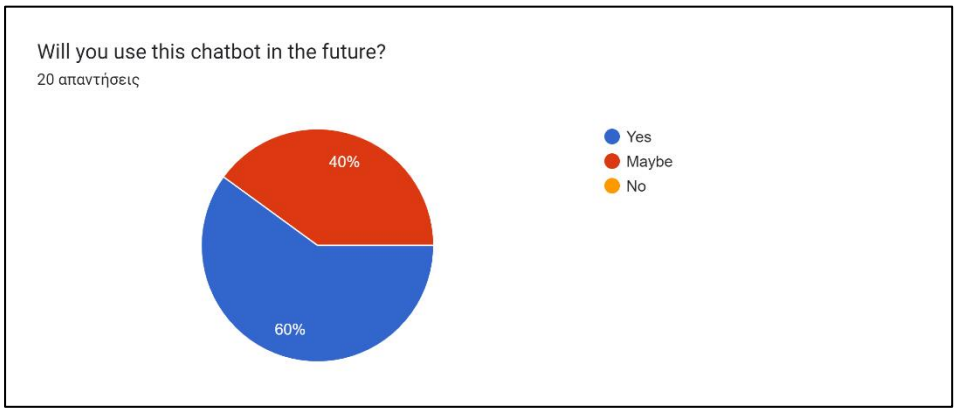
20 απαντήσεις

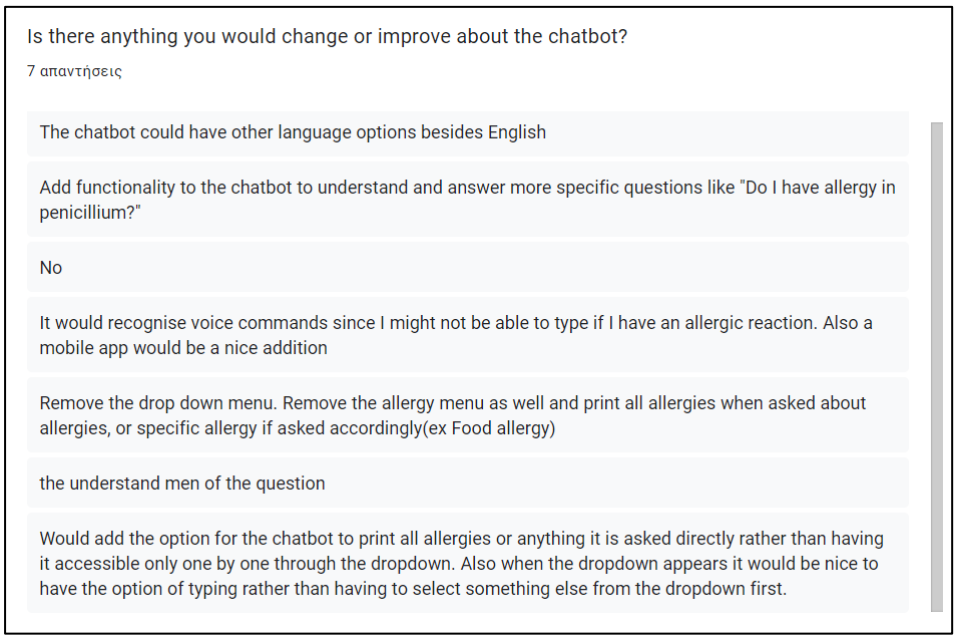


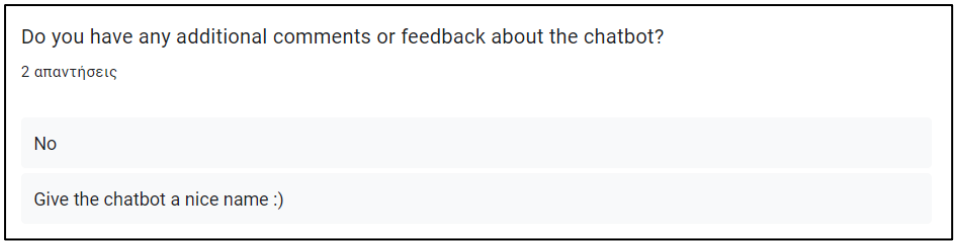
How satisfied were you with your overall experience using the chatbot?

20 απαντήσεις



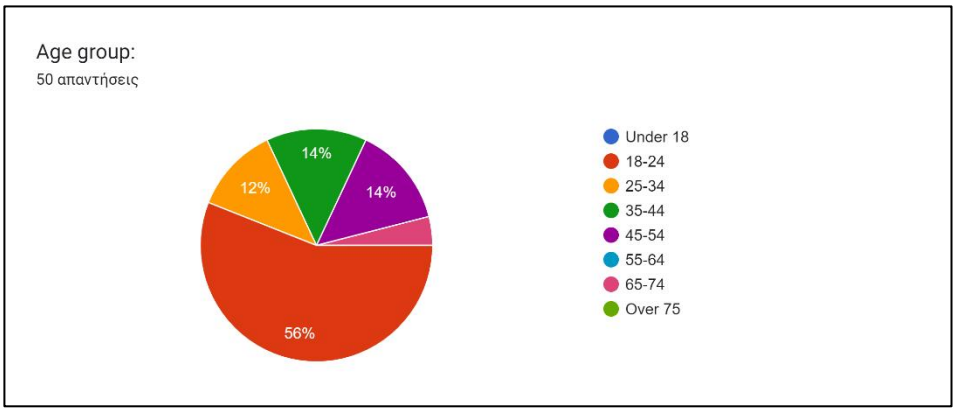


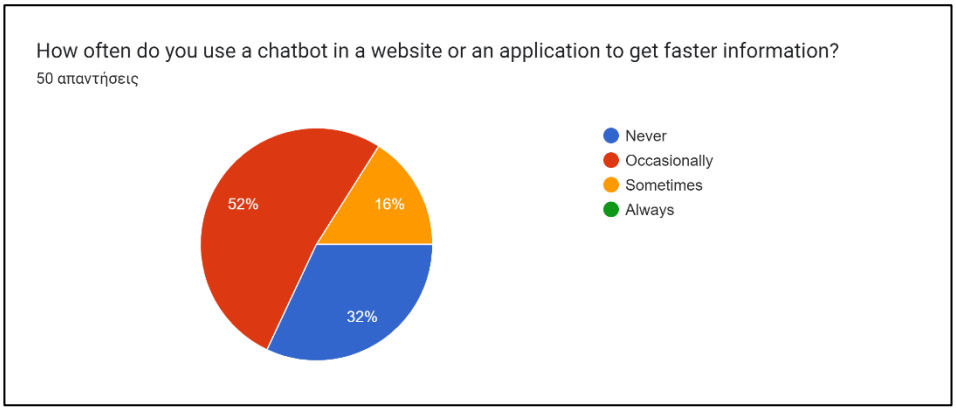


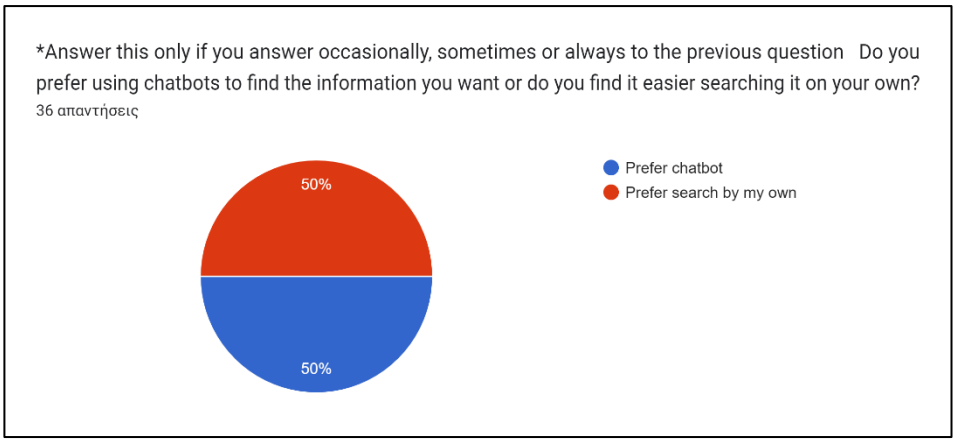


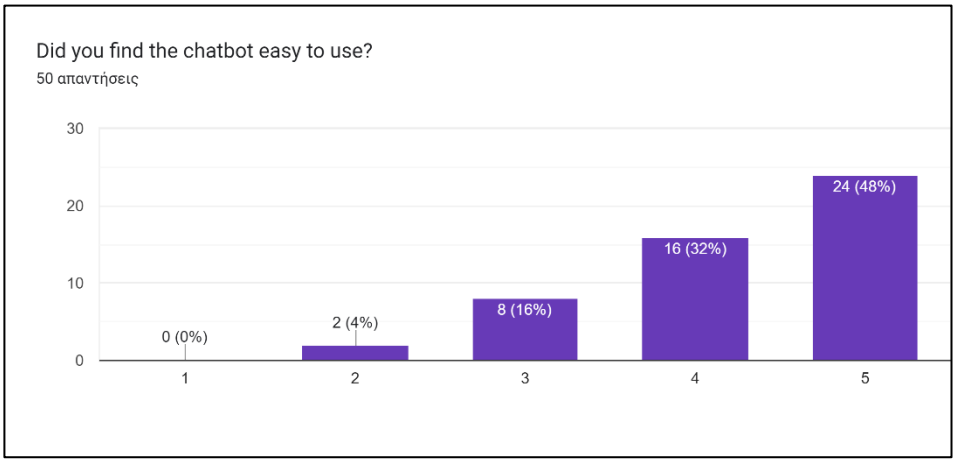
Από τις απαντήσεις που συλλέχθηκαν και φαίνονται πιο πάνω, προέκυψε η ανάγκη για μικρές τροποποιήσεις του chatbot ώστε να επιλυθούν κάποια από τα προβλήματα που παρουσιάστηκαν κατά την πρώτη φάση της δοκιμής. Τα γενικά συμπεράσματα και η μελλοντική εργασία παρουσιάζονται και αναλύονται στη συνέχεια.

6.4.2 Αποτελέσματα Δεύτερης Φάσης



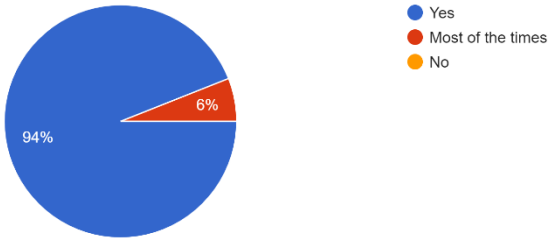






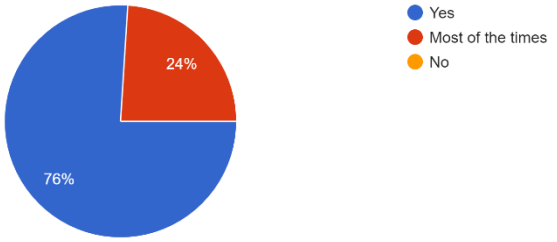
Did the chatbot respond quickly to your questions?

50 απαντήσεις



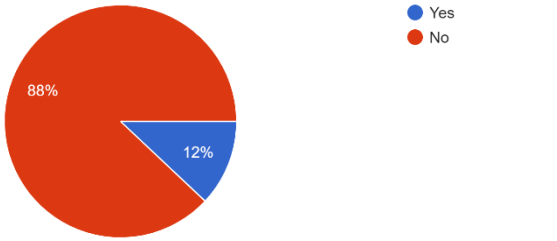
Did the chatbot provide clear and understandable answers?

50 απαντήσεις



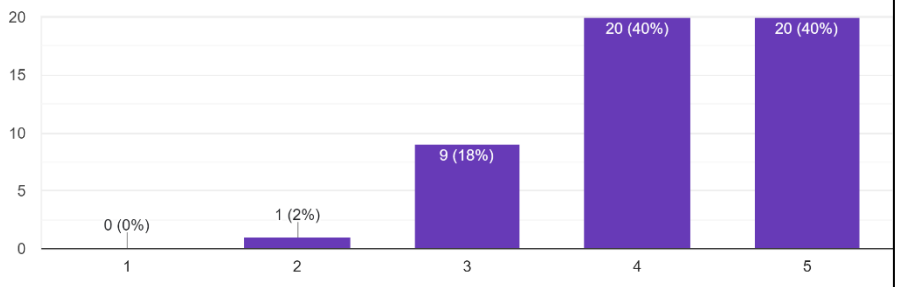
Did you encounter any errors or bugs while using the chatbot?

50 απαντήσεις



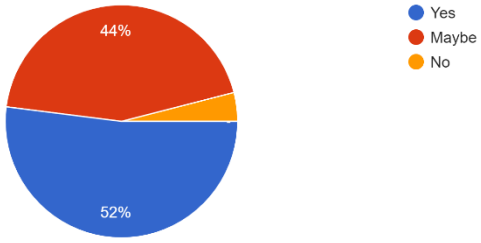
How satisfied were you with your overall experience using the chatbot?

50 απαντήσεις



Will you use this chatbot in the future?

50 απαντήσεις



Is there anything you would change or improve about the chatbot?

18 απαντήσεις

More information such as Who is my doctor?

It would be great if the chatbot was able to connect the perscriptions for medicine to health problems, essentially helping the users recognize which drugs they are taking and which problem they help with. Also, it would be great for usability for drop down menus to not make the user unable to use the keyboard and perhaps be able to choose an answer from the drop down menu using his keyboard as well. In case the chatbot doesn't understand the user's answer, I would change the answer that is given to a clear "Sorry, I didn't understand that" rather than an answer that would be given to another question. In a later stage, being able to use speech recognition to write to the chatbot would make it even easier to use.

Better NLP

I would like to ask if any current/new symptoms is related to any illnesses/disease I have/had (e.g. autoimmune disease)

Have the option to allow the user to click on a request instead of only typing it

Addition of synonym keywords, word stem, and common spelling mistakes, If I say Nothing on the dropdown, I would expect to continue to write, or make sure I want to end the conversation with a confirmation prompt, It doesn't seem to be conversational its more like a search which can be more than enough for this application., I would make the list use optional in case the bot misunderstood, so I don't have to press something else., If I ask for its name it logs out,

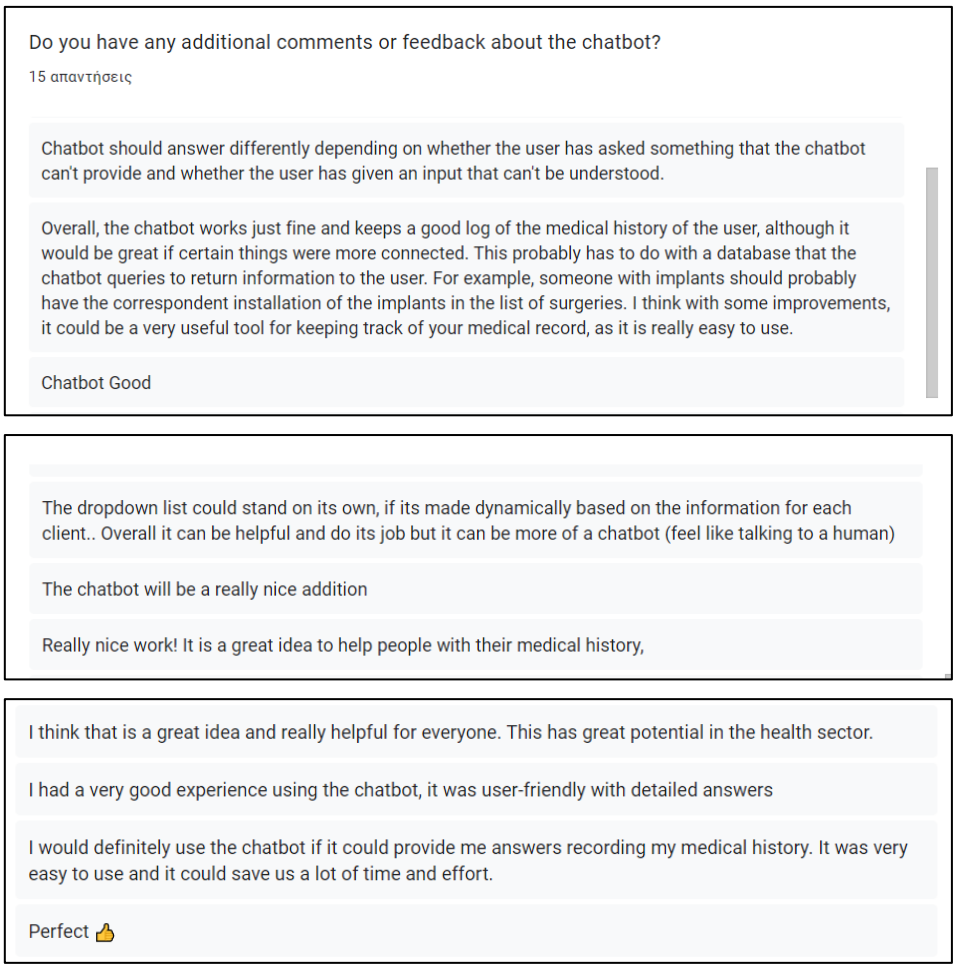
Maybe it can use AI to provide more information and the responses can be more user friendly

Yes

I would ameliorate its capabilities and intelligence. It couldn't understand more complex questions on the categories. It could only just read the database and list the contents, which is not what chatbots are used for. These informations could instead be read from a table or a list where the patient would have access.

I would add the result of a surgery they might have done (or any complications) and more information about their prescription (e.g. what is it for, written simply so the patient can understand it easily)

Nothing



6.5 Συμπεράσματα Δοκιμής

Κατά τις δυο φάσεις δοκιμής δοκίμασαν και απάντησαν το ερωτηματολόγιο συνολικά 46 άτομα. Τα άτομα αυτά ήταν διαφόρων ηλικιών (από 18 μέχρι και 74), η πλειοψηφία, όμως, ανήκε στην ηλικιακή ομάδα 18 με 24 έτη (56%).

Ξεκινώντας, στις γενικές ερωτήσεις που αφορούσαν τη συλλογή δεδομένων σχετικά με το αν οι συμμετέχοντες χρησιμοποιούν ή όχι chatbots στην καθημερινότητα τους και αν τα προτιμούν για την εύρεση πληροφοριών, παρατηρήθηκε ότι στην πλειοψηφία τους οι συμμετέχοντες είτε δεν χρησιμοποιούν καθόλου chatbots (32%) είτε τα χρησιμοποιούν περιστασιακά (52%). Επιπλέον, παρατηρήθηκε ότι κανένας από τους συμμετέχοντες δεν χρησιμοποιεί chatbot για να έχει γρήγορη πρόσβαση σε πληροφορίες. Από τα άτομα που απάντησαν ότι χρησιμοποιούν περιστασιακά ή κάποτε τα chatbots οι μισοί περίπου απάντησαν ότι τα προτιμούν για να έχουν γρήγορη πρόσβαση σε πληροφορίες, ενώ οι άλλοι μισοί ότι προτιμούν να ψάχνουν μόνοι τους τις πληροφορίες που θέλουν.

Από τα πιο πάνω δεδομένα, συμπεραίνουμε ότι πολύς κόσμος, ακόμη και σήμερα που τα chatbots και οι virtual assistance ξεκίνησαν να εμφανίζονται σχεδόν σε όλες τις ιστοσελίδες, δεν είναι εξοικειωμένοι με τη χρήση τους ή προτιμούν να μην τα χρησιμοποιούν.

Προχωρώντας στην αξιολόγηση του παρόντος συστήματος, παρατηρήθηκε πως η ιδέα για την υλοποίηση chatbot για εύκολη πρόσβαση σε ιατρικό ιστορικό, αποτελεί μια αρκετά καλή ιδέα η οποία θα μπορούσε να διευκολύνει τόσο τους ασθενείς όσο και τους παρόχους υγείας.

Πιο συγκεκριμένα, το chatbot,όπως παρατηρήθηκε από τα δεδομένα που συλλέχθηκαν, είναι αρκετά εύκολο στη χρήση με την πλειοψηφία να απαντά πως η χρήση του chatbot του φάνηκε εύκολη (4-easy) ή πολύ εύκολη (5- really easy)- ποσοστά 32% και 48% αντίστοιχα. Επιπρόσθετα, οι συμμετέχοντες θεωρούν ότι το chatbot ανταποκρινόταν γρήγορα, δηλαδή έδινε απαντήσεις σε μικρό χρονικό διάστημα (94%) και τέλος οι απαντήσεις που έδινε ήταν πάντα ή τις περισσότερες φορές ακριβείς και κατανοητές (ποσοστά 76% και 24% αντίστοιχα). Στην ερώτηση που αφορούσε την αξιολόγηση της γενικής εμπειρίας που είχαν οι συμμετέχοντες με τη χρήση του chatbot, οι περισσότεροι έμειναν ικανοποιημένοι. Συγκεκριμένα, η πλειοψηφία των συμμετεχόντων έμειναν αρκετά ή πολύ ικανοποιημένοι χρησιμοποιώντας το chatbot (ποσοστά 40% και 40% αντίστοιχα) και απάντησαν πως θα χρησιμοποιούσαν το εν λόγω chatbot στο μέλλον αν τους δινόταν αυτή η δυνατότητα.

Όσο αφορά το αν εμφανίστηκαν προβλήματα κατά τη δοκιμή, παρατηρήθηκε πως errors και bugs υπήρχαν, εντοπίστηκαν και διορθώθηκαν κατά την πρώτη φάση της δοκιμής, ενώ στη δεύτερη φάση της δοκιμής κανένας από τους συμμετέχοντες δεν ανέφερε κάποιο error ή bug. Αυτό σημαίνει ότι οι διορθώσεις που πραγματοποιήθηκαν μεταξύ των δύο φάσεων μείωσαν τα σφάλματα σχεδόν στο ελάχιστο, αν όχι τελείως. Errors και bugs θεωρούνται προβλήματα όπως το chatbot να μην επεξεργάζεται την ερώτηση και να σταματά ή να εμφανίζεται error message πράγμα που σημαίνει ότι για κάποιο λόγο το chatbot σταμάτησε τη ροή συζήτησης. Οι λανθασμένες απαντήσεις ή η μη κατανόηση μιας ερώτησης δεν θεωρείται error ή bug.

Κατά τη λήψη δεδομένων, καταγράφηκαν σαν σφάλματα και λανθασμένες απαντήσεις του chatbot. Αυτού του είδους λάθη, δεν αφορούν πραγματικό error ή bug, όπως ήδη αναφέρθηκε πιο πάνω, αλλά λήφθηκαν υπόψιν και έγινε μια προσπάθεια για την επανεκπαίδευση και προσαρμογή του NLU engine με σκοπό να μειωθούν στο ελάχιστό οι λανθασμένες απαντήσεις που δίνει το chatbot.

Τέλος, μέσω του ερωτηματολόγιου συλλέχθηκαν γενικά σχόλια για το chatbot και εισηγήσεις για μελλοντική βελτίωση του ή για προσθήκη περισσότερων δυνατοτήτων. Αυτό που παρατηρήθηκε είναι πως οι συμμετέχοντες θα ήθελαν το chatbot να αναγνωρίζει και να είναι σε θέση να απαντά σε πιο περίπλοκες και πιο συγκεκριμένες ερωτήσεις, όπως για παράδειγμα να είναι σε θέση να απαντά ναι ή όχι σε ερωτήσεις του τύπου «Do I have allergy in penicillium?» , «I am vaccinated for Covid-19?» και τα λοιπά. Επίσης, υπάρχουν εισηγήσεις για αφαίρεση των dropdown λιστών, για αναγνώριση φωνητικών μηνυμάτων και για προσθήκη άλλων γλωσσών. Όλα τα σχόλια λήφθηκαν υπόψη και οργανώθηκαν ολοκληρωμένες ιδέες και εισηγήσεις για μελλοντική αναβάθμιση του συστήματος οι οποίες παρουσιάζονται στο Κεφάλαιο 7.

Συμπερασματικά, η υλοποίηση και η προσθήκη ενός chatbot για τη διευκόλυνση εύρεσης δεδομένων που αφορούν το Patient Summary και e-Prescription κάθε ατόμου, φαίνεται να είναι μια χρήσιμη και αποτελεσματική ιδέα. Από την ανατροφοδότηση που συλλέχθηκε, ένα τέτοιο chatbot θα βοηθούσε σε μεγάλο βαθμό και ασθενείς και παρόχους υγείας ώστε να υπάρχει γρήγορη υγειονομική περίθαλψη και πληροφόρηση.

Κεφάλαιο 7 – Μελλοντική Εργασία

7.1 Μελλοντική Εργασία

Μετά τη χρήση του chatbot κατά τη διάρκεια της δοκιμής και την ανατροφοδότηση που έδωσαν οι χρήστες, τα σχόλια και οι εισηγήσεις τους οργανώθηκαν σε συγκεκριμένες προτάσεις για μελλοντική αναβάθμιση ή επιπλέον προσθήκες στο σύστημα που αναπτύχθηκε. Οι ιδέες αυτές παρουσιάζονται πιο κάτω:

1. **Προσθήκη επιπλέον γλωσσών:** Το chatbot στην παρούσα φάση αντιλαμβάνεται ερωτήσεις και δίνει απαντήσεις μόνο στην αγγλική γλώσσα. Σαν μελλοντική προσθήκη θα μπορούσε να δημιουργηθεί η δυνατότητα επιλογής της γλώσσας επικοινωνίας ώστε κάθε χρήστης να μπορεί να επικοινωνήσει με το chatbot σε όποια γλώσσα αυτός προτιμά.

2. **Αναγνώριση φωνητικών εντολών:** Αυτή την στιγμή η επικοινωνία με το chatbot γίνεται μόνο μέσω γραπτών μηνυμάτων. Στο μέλλον θα μπορούσε να υπάρχει δυνατότητα επικοινωνίας και μέσω φωνητικών μηνυμάτων, έτσι ώστε να μπορούν να το χρησιμοποιούν και άτομα με προβλήματα όρασης ή άλλα προβλήματα που εμποδίζουν τη γραπτή επικοινωνία.

3. **Αναγνώριση πιο περίπλοκων και πιο συγκεκριμένων ερωτήσεων:** Το chatbot είναι σε θέση να αναγνωρίζει βασικές λέξεις κλειδιά και να αποφασίζει την κατηγορία που ανήκει η ερώτηση του χρήστη, όπως επεξηγήθηκε σε προηγούμενο κεφάλαιο. Δεν μπορεί, όμως, να αναγνωρίσει ερωτήσεις που ζητούν συγκεκριμένη πληροφορία. Για παράδειγμα, αν ο χρήστης ρωτήσει "Do I have allergy in penicill?", το chatbot θα τον οδηγήσει στην κατηγορία allergies και δεν θα απαντήσει Yes ή No. Σαν μελλοντική αναβάθμιση θα μπορούσε να δομηθεί εκ νέου το NLU engine και να εκπαιδευτεί ξανά το chatbot ώστε να είναι σε θέση να αντιλαμβάνεται και πιο περίπλοκες και συγκεκριμένες ερωτήσεις.

4. **Παροχή επιπλέον πληροφοριών:** Το patient summary σαν έγγραφο περιλαμβάνει και άλλες πληροφορίες πέρα αυτών που αναγνωρίζει την παρούσα στιγμή το chatbot. Όπως αναφέρθηκε και πιο πάνω, το chatbot δίνει πληροφορίες για αλλεργίες, εμβολιασμό, ιατρικές συσκευές και εμφυτεύματα, ασθένειες, τρέχοντα προβλήματα, επεμβάσεις και φαρμακευτική αγωγή. Θα μπορούσε, όμως, να παρέχει επιπλέον πληροφορίες. Για παράδειγμα θα μπορούσε να δίνει στον χρήστη τη δυνατότητα να έχει πρόσβαση σε προσωπικές πληροφορίες, την ομάδα αίματος, ιστορικό εγκυμοσύνης και τα λοιπά.

5. **Αφαίρεση ή αναπροσαρμογή του dropdown list των αλλεργιών:** Από την ανατροφοδότηση που δόθηκε, πολλοί από τους χρήστες εισηγήθηκαν να μην υπάρχει ολόκληρο το dropdown list των αλλεργιών αλλά το chatbot να επιστρέφει λίστα μόνο με τις κατηγορίες των αλλεργιών που έχει ο εκάστοτε χρήστης. Αυτό θα εξοικονομούσε χρόνο αφού δεν θα χρειαζόταν ο κάθε χρήστης να ελέγχει μια προς μια όλες τις κατηγορίες για να δει τις αλλεργίες που παρουσιάζει. Μια άλλη εισήγηση που δόθηκε ήταν το dropdown list να αφαιρεθεί εντελώς και να παρουσιάζονται κατευθείαν όλες οι αλλεργίες του χρήστη όπως γίνεται και στις άλλες κατηγορίες.

Βιβλιογραφία

[1] AppAdvice. (n.d.). Vik Breast. Retrieved from https://appadvice.com/app/vik-breast/1446263668

[2] Babylon Health. (n.d.). Babylon Health. Retrieved from https://www.babylonhealth.com/

[3] BMC Software. (n.d.). NLU vs NLP: What's the Difference? Retrieved from https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/

[4] Boost.ai. (n.d.). What is conversational AI, anyway? Retrieved from https://www.boost.ai/knowledge/what-is-conversational-ai

[5] BotPress documentation. Retrieved from http://botpress-docs.s3-website-us-east-1.amazonaws.com/docs/11.9.6/main/overview/

[6] Botpress. (n.d.). Botpress v12. Retrieved from https://v12.botpress.com/overview/quickstart/conversation-studio

[7] Botpress. (n.d.). Botpress: The open source conversational platform. Retrieved from https://botpress.com/

[8] Centers for Medicare & Medicaid Services. (n.d.). E-Prescribing. Retrieved from https://www.cms.gov/Medicare/E-Health/Eprescribing

[9] Connolly, E. (n.d.). Principles of bot design Retrieved from https://www.intercom.com/blog/principles-bot-design/

[10] Contributors to Wikimedia projects. (2022, February 17). Turing test. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Turing_test

[11] Contributors to Wikimedia projects. (2022, March 18). Electronic prescribing. In Wikipedia, The Free Encyclopaedia. Retrieved from https://en.wikipedia.org/wiki/Electronic_prescribing

[12] DeepPavlov. (n.d.). DeepPavlov: An open source conversational AI framework. Retrieved from https://deeppavlov.ai/

[13] Earle, N. (n.d.). GitHub Gist: 80150ff1c50031e59b872baf0e474977. Retrieved from https://gist.github.com/NigelEarle/80150ff1c50031e59b872baf0e474977

[14] European Parliament. (2020, August 27). Τι είναι η τεχνητή νοημοσύνη και πώς χρησιμοποιείται. Retrieved from https://www.europarl.europa.eu/news/el/headlines/society/20200827STO85804/ti-einai-i-techniti-noimosuni-kai-pos-chrisimopoieitai

[15] Hamilton, N. (2008, July 31). The A-Z of Programming Languages: JavaScript. Computerworld. Retrieved from

https://www.computerworld.com.au/article/254378/-z_programming_languages_javascript/

[16] IBM. (n.d.). Artificial Intelligence. Retrieved from https://www.ibm.com/topics/artificial-intelligence

[17] IBM. (n.d.). Natural Language Processing. Retrieved from https://www.ibm.com/topics/natural-language-processing

[18] IP.GR. (n.d.). Τι είναι η HTML 6.1? Retrieved from https://www.ip.gr/Web_Development/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%B7-html-61.html

[19] IP.GR. (n.d.). Τι είναι το CSS 3.27? Retrieved from https://www.ip.gr/Web_Development/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CF%84%CE%BF-css-327.html

[20] Joint Initiative Council for Global Health Informatics Standardization. (n.d.). The International Patient Summary key health data, worldwide. Retrieved from https://international-patient-summary.net/

[21] Kandola, A. (2019, August 23). STORY: ELIZA, The First Chatbot Developed In 1966. Analytics India Magazine. Retrieved from https://analyticsindiamag.com/story-eliza-first-chatbot-developed-1966/

[22] Kapitsaki, G. (n.d.). ΕΠΛ343: Τεχνολογίες Λογισμικού [Course lectures]. Department of Computer Science, University of Cyprus. Retrieved from https://www.cs.ucy.ac.cy/courses/EPL343/index.html

[23] Kohavi, R., & Provost, F. (1998). Glossary of terms. Retrieved from https://www.researchgate.net/publication/220211726_Glossary_of_Terms

[24] Krill, P. (2008, June 23). JavaScript creator ponders past, future. InfoWorld. Archived from the original on March 30, 2009. Retrieved May 19, 2009 from https://www.infoworld.com/article/2653227/javascript-creator-ponders-past--future.html

[25] Landsiedel, J. (n.d.). What is NLP? Retrieved from https://www.landsiedel.com/en/nlp/what-is-nlp.html

[26] Lybrate. (n.d.). Lybrate. Retrieved from https://www.lybrate.com/

[27] Netscape Communications Corp. (1995, December 4). Netscape And Sun Announce JavaScript, The Open, Cross-Platform Object Scripting Language For Enterprise Networks And The Internet [Press release]. Archived from the original on 2007-09-16.

https://web.archive.org/web/20070916144913/https://wp.netscape.com/newsref/pr/newsrelease67.html

[28] Oracle Corporation. (2020). MySQL 8.0 reference manual. Retrieved April 3, 2023, from https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[29] Ozanich, A. (2022, February 09). What Are JSON Files & How Do You Use Them? Retrieved from https://blog.hubspot.com/website/json-files

[30] PyTorch. (n.d.). PyTorch: An open source machine learning framework. Retrieved from https://pytorch.org

[31] Rasa. (n.d.). Rasa: Open source conversational AI. Retrieved from https://rasa.com/

[32] Scale Nut. (n.d.). NLP vs NLU. Retrieved from https://www.scalenut.com/blogs/nlp-vs-nlu

[33] Simon, P. (2013). Too Big to Ignore: The Business Case for Big Data. John Wiley & Sons.

[34] The NLP Academy. (n.d.). What is NLP? Retrieved from https://www.nlpacademy.co.uk/what_is_nlp/

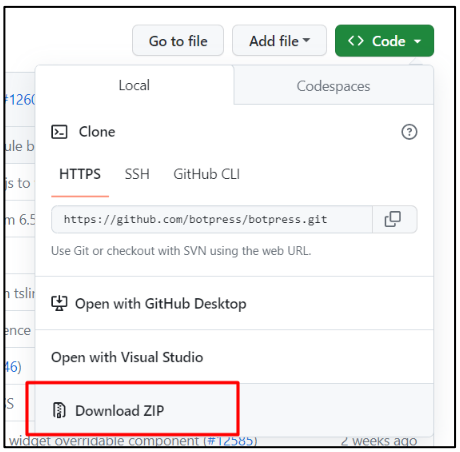
[35] Weizenbaum, J. (1966). ELIZA - A computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45. Retrieved from https://web.njit.edu/~ronkowit/eliza.html

[36] Zabój, D. (n.d.). 8 Conversation Design Principles for Making Better Chatbot Conversations Retrieved from https://www.chatbot.com/blog/conversational-design/

Παράρτημα Α: Εγκατάσταση BotPress server

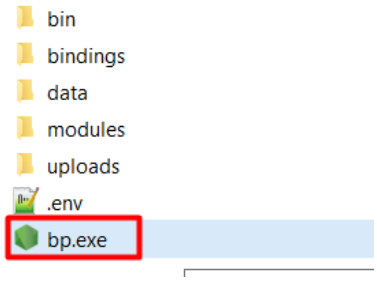
Η εγκατάσταση και ο τρόπος λειτουργίας του BotPress server είναι αρκετά απλός [5].

1. Κατεβάζουμε το zip αρχείο του BotPress από το GitHub (https://github.com/botpress/botpress.git)



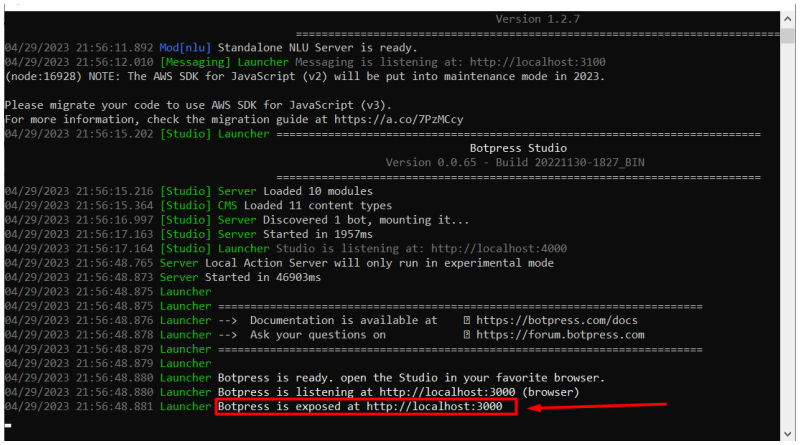
2. Κάνουμε extract το αρχείο στο directory που επιθυμούμε ή δημιουργούμε ένα καινούργιο directory για να το αποθηκεύσουμε.

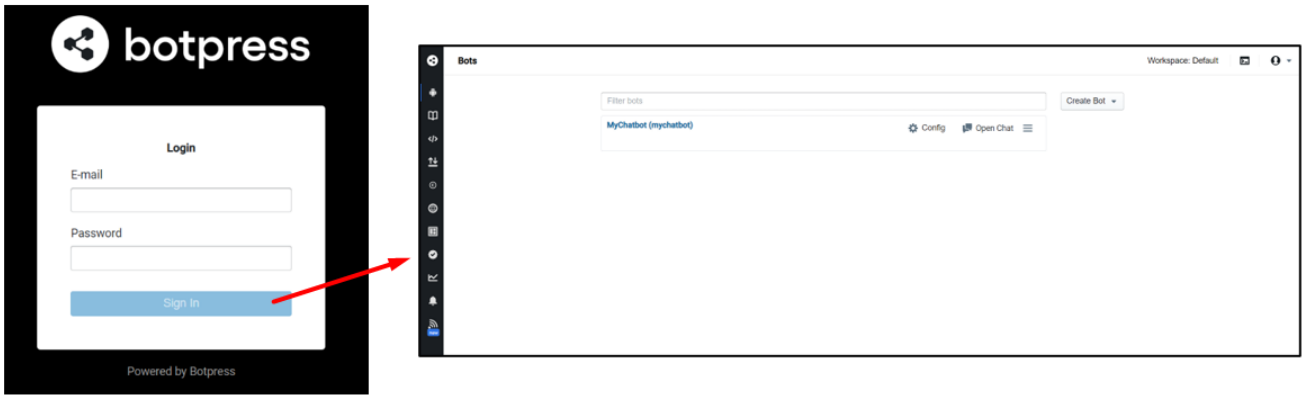
3. Τρέχουμε το .exe file που εμφανίζεται.



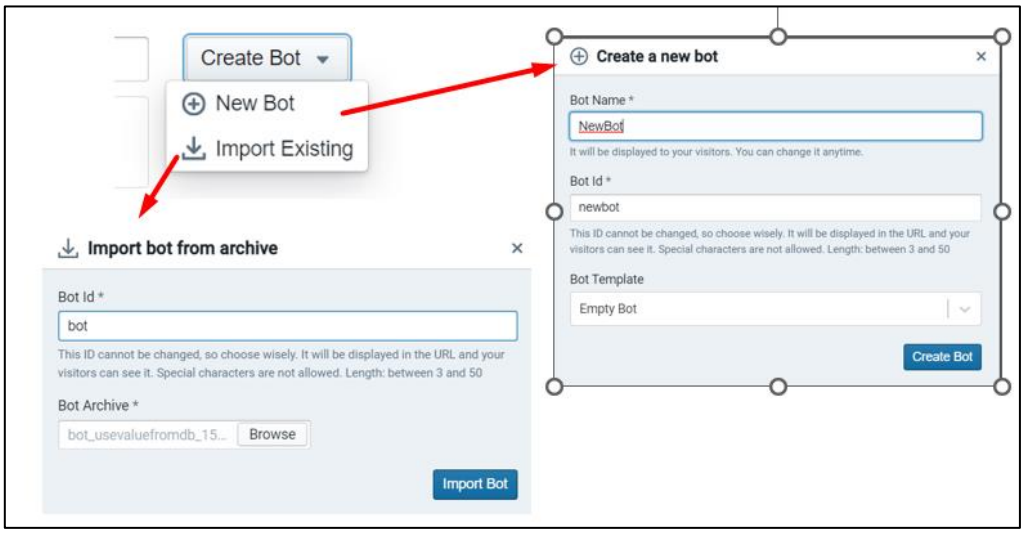
Με αυτό τον τρόπο ο BotPress server τίθεται σε λειτουργία και μπορούμε να χτίσουμε ένα chatbot ή να ανεβάσουμε κάποιο ήδη υλοποιημένο chatbot, ώστε να μπορούμε να το τρέξουμε ή να το τροποποιήσουμε.

Αν δεν αλλάξουμε τα settings, τότε το admin panel αυτόματα γίνεται exposed στο localhost:3000, μέσω του οποίου μπορείτε να ενωθείτε και να επεξεργαστείτε τα chatbot τα οποία έχετε ή να δημιουργήσετε κάποιο καινούργιο. (Σημείωση: Μπορείτε ανάλογα με το τι επιθυμείτε να κάνετε το botpress studio να γίνεται exposed σε οποιοδήποτε URL θέλετε.)

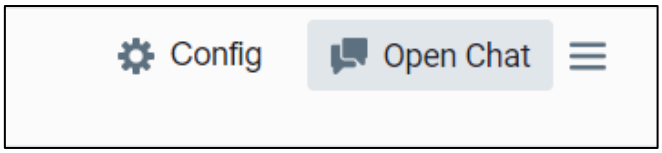




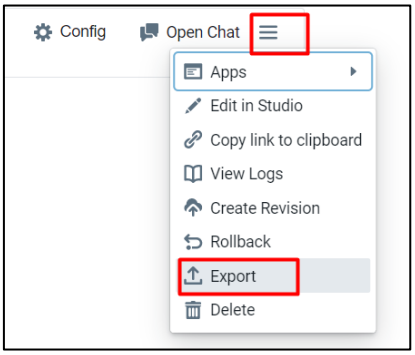
Αν επιθυμείτε να ανεβάσετέ ήδη υλοποιημένο chatbot, πηγαίνετε στο Create Bot και επιλέξτε import existing. Ενώ αν θέλετε να δημιουργήσετε καινούργιο, επιλέξτε New Bot. Για να γίνει import ένα chatbot πρέπει να είναι zip αρχείο που δημιουργήθηκε βάσει των προδιαγραφών από κάποιο άλλο server botpress.



Μέσω του OpenChat έχετε πρόσβαση στη web μορφή του chatbot. By default το web chat είναι exposed στο localhost:3000/lite/mychatbot/…



Για να μεταφέρετε ένα bot σε κάποιον άλλο BotPress server, το μόνο που πρέπει να κάνετε είναι export το chatbot και ακολούθως import στον άλλο server.



Παράρτημα Β: Flows

- **main.flow.json**


```
{
  "version": "0.0.1",
  "catchAll": {
    "onReceive": [],
    "next": []
  },
  "startNode": "entry",
  "nodes": [
    {
      "id": "entry",
      "name": "entry",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "Need-help-choice"
        }
      ],
      "onEnter": [
        "say #!builtin_text-via-9l"
      ],
      "onReceive": null
    },
    {
      "id": "skill-49c646",
      "type": "skill-call",
      "skill": "choice",
      "name": "Need-help-choice",
      "flow": "skills/choice-49c646.flow.json",
      "next": [
        {
          "caption": "User picked [yes]",
          "condition": "temp['skill-choice-ret-ufb10dsalv'] == \"yes\"",
          "node": "log-in"
        },
        {
          "caption": "User picked [no]",
          "condition": "temp['skill-choice-ret-ufb10dsalv'] == \"no\"",
          "node": "END"
        },
        {
          "caption": "On failure",
          "condition": "true",
          "node": "error.flow.json"
        }
      ],
      "onEnter": null,
```

```
      "onReceive": null
    },
    {
      "id": "d0d5aab690",
      "name": "log-in",
      "onEnter": [
        "say #!builtin_text-gV1jia"
      ],
      "onReceive": [
        "builtin/setVariable
{\"type\":\"session\",\"name\":\"userId\",\"value\":\"{{event.preview}}\"}"
      ],
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "before-login-help"
        },
        {
          "condition": "isNaN(session.userId)",
          "conditionType": "raw",
          "node": "wrong-id"
        },
        {
          "condition": "session.userId != null",
          "conditionType": "raw",
          "node": "check-for-id-in-database"
        },
        {
          "condition": "true",
          "conditionType": "always",
          "node": "Misunderstand"
        }
      ],
      "type": "standard"
    },
    {
      "id": "75d7dc74ab",
      "name": "check-for-id-in-database",
      "next": [
        {
          "condition": "user.response == \"yes\"",
          "conditionType": "raw",
          "node": "Welcome-message"
        },
        {
          "condition": "user.response == \"no\"",
          "conditionType": "raw",
          "node": "wrong-id"
        }
```

```
      ],
      "onEnter": [
        "exist-check
{\"table\":\"personal_info\",\"key\":\"p_name\",\"whereColumn\":\"p_id\",\"value\"
:\"{{session.userId}}\",\"database\":\"patients_info\",\"hash\":\"0\"}"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "9cec62413c",
      "name": "wrong-id",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "log-in"
        }
      ],
      "onEnter": [
        "say #!builtin_text-2XLqs1"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "d39501d664",
      "name": "before-login-help",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "Id-help"
        }
      ],
      "onEnter": [
        "say #!builtin_text-mXBIyT"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "db6f8d818f",
      "name": "questions_anwsers",
      "onEnter": [],
      "onReceive": [],
      "next": [
        {
          "condition": "event.nlu.slots.drugAllergies.value != undefined",
          "conditionType": "raw",
```

```
      "node": "allergies.flow.json#name-the-drug-allergies"
    },
    {
      "condition": "event.nlu.slots.foodAllergies.value != undefined",
      "conditionType": "raw",
      "node": "allergies.flow.json#name-the-food-allergies"
    },
    {
      "condition": "event.nlu.slots.drugIntolerances.value != undefined",
      "conditionType": "raw",
      "node": "allergies.flow.json#name-the-drug-intolerance"
    },
    {
      "condition": "event.nlu.slots.foodIntolerances.value != undefined",
      "conditionType": "raw",
      "node": "allergies.flow.json#name-the-food-intolerance"
    },
    {
      "condition": "event.nlu.slots.substanceAllergies.value != undefined",
      "conditionType": "raw",
      "node": "allergies.flow.json#name-the-substance-allergies"
    },
    {
      "condition": "event.nlu.intent.name === 'allergies'",
      "conditionType": "intent",
      "node": "allergies.flow.json"
    },
    {
      "condition": "event.nlu.intent.name === 'vaccinations'",
      "conditionType": "intent",
      "node": "vaccinations.flow.json"
    },
    {
      "condition": "event.nlu.intent.name === 'past_illnesses'",
      "conditionType": "intent",
      "node": "past_illnesses.flow.json"
    },
    {
      "condition": "event.nlu.intent.name === 'surgeries'",
      "conditionType": "intent",
      "node": "surgeries.flow.json"
    },
    {
      "condition": "event.nlu.intent.name === 'medical_devices'",
      "conditionType": "intent",
      "node": "medical_devices_implants.flow.json"
    },
    {
      "condition": "event.nlu.intent.name === 'current_problems'",
      "conditionType": "intent",
```

```
        "node": "current_problems.flow.json"
      },
      {
        "condition": "event.nlu.intent.name === 'prescription_summary'",
        "conditionType": "intent",
        "node": "prescription_summary.flow.json"
      },
      {
        "condition": "event.nlu.intent.name === 'help'",
        "conditionType": "intent",
        "node": "help.flow.json"
      },
      {
        "condition": "event.nlu.intent.name === 'log_out'",
        "conditionType": "intent",
        "node": "end_flow.flow.json"
      },
      {
        "condition": "true",
        "conditionType": "always",
        "node": "misunderstand"
      }
    ],
    "type": "standard"
  },
  {
    "id": "6b6c0b92ed",
    "name": "Misunderstand",
    "next": [
      {
        "condition": "true",
        "node": "log-in"
      }
    ],
    "onEnter": [
      "say #!builtin_text-gU1sSN"
    ],
    "onReceive": null,
    "type": "standard"
  },
  {
    "id": "40467c6ce5",
    "name": "Id-help",
    "next": [
      {
        "condition": "true",
        "conditionType": "always",
        "node": "log-in"
      }
    ],
```

```
      "onEnter": [
        "say #!builtin_text-wzkIzH"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "043013982d",
      "name": "no-medical-data",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "end_flow.flow.json"
        }
      ],
      "onEnter": [
        "say #!builtin_text-fY3Lba",
        "say #!builtin_text-_at2aC"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "da02d96a59",
      "name": "AES-encryprion",
      "next": [
        {
          "condition": "user.response == \"yes\"",
          "conditionType": "raw",
          "node": "ask-questions"
        },
        {
          "condition": "true",
          "conditionType": "always",
          "node": "no-medical-data"
        }
      ],
      "onEnter": [
        "AES_encryption
{\"id\":\"{{session.userId}}\",\"date\":\"{{user.data}}\"}",
        "exist-check
{\"table\":\"patient_data\",\"key\":\"id\",\"whereColumn\":\"patient_hash\",\"valu
e\":\"{{user.hash}}\",\"database\":\"myDatabase\",\"hash\":\"1\"}",
        "builtin/setVariable
{\"type\":\"session\",\"name\":\"userId\",\"value\":\"{{user.data}}\"}"
      ],
      "onReceive": null,
      "type": "standard"
    },
```

```
    {
      "id": "0c566706c6",
      "name": "misunderstand",
      "next": [
        {
          "condition": "true",
          "node": "ask-questions"
        }
      ],
      "onEnter": [
        "say #!builtin_text-Gh_NfH"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "23bb128cce",
      "name": "Welcome-message",
      "next": [
        {
          "condition": "user.response == \"yes\"",
          "conditionType": "raw",
          "node": "AES-encryprion"
        },
        {
          "condition": "true",
          "conditionType": "always",
          "node": "no-medical-data"
        }
      ],
      "onEnter": [
        "say #!builtin_text-ev-aZ0",
        "exist-check
{\"table\":\"personal_info\",\"key\":\"DATE_FORMAT(p_birth_date, '%Y-%m-
%d')\",\"whereColumn\":\"p_id\",\"value\":\"{{session.userId}}\",\"database\":\"pa
tients_info\",\"hash\":\"0\"}"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "f9eed285ae",
      "name": "ask-questions",
      "next": [
        {
          "condition": "true",
          "node": "questions_anwsers"
        }
      ],
      "onEnter": [
```

```
        "say #!builtin_text-rt_-9m"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **allergies.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "0095d300cf",
      "name": "entry",
      "onEnter": [
        "exist-check
{\"table\":\"allergies_mapping\",\"key\":\"count(allergy_coding_id)\",\"whereColum
n\":\"patient_id\",\"value\":\"{{session.userId}}\",\"hash\":\"0\",\"database\":\"
myDatabase\"}"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "user.data == 0",
          "conditionType": "raw",
          "node": "no-allergies"
        },
        {
          "condition": "true",
          "node": "Count-allergies"
        }
      ],
      "type": "standard"
    },
    {
      "id": "1823fe7ce1",
      "name": "Count-allergies",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
```

```
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "conditionType": "always",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "say #!builtin_text-WSMSku"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "e8bec9f7b6",
      "name": "name-the-food-allergies",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"allergy\",\"id\":\"{{session.userId}}\",\"category\":\"Food
allergy\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "skill-e5b13d",
      "type": "skill-call",
      "skill": "choice",
      "name": "more-details",
      "flow": "skills/choice-e5b13d.flow.json",
      "next": [
        {
```

```
          "caption": "User picked [Go back]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Go back\"",
          "conditionType": "raw",
          "node": "main.flow.json#ask-questions"
        },
        {
          "caption": "User picked [Food al...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Food
allergies\"",
          "node": "name-the-food-allergies"
        },
        {
          "caption": "User picked [Allergy...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Allergy to
substances\"",
          "node": "name-the-substance-allergies"
        },
        {
          "caption": "User picked [Drug al...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Drug
allergies\"",
          "node": "name-the-drug-allergies"
        },
        {
          "caption": "User picked [Drug in...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Drug
intolerances\"",
          "node": "name-the-drug-intolerance"
        },
        {
          "caption": "User picked [Food in...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Food
intolerances\"",
          "node": "name-the-food-intolerance"
        },
        {
          "caption": "User picked [Intoler...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Intolerances\"",
          "node": "name-the-intolerance"
        },
        {
          "caption": "User picked [Propens...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Propensity to
adverse reactions to drug\"",
          "node": "name-the-propensity-to-drug"
        },
        {
          "caption": "User picked [Propens...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Propensity to
adverse reactions to food\"",
```

```
          "node": "name-the-propensity-to-food"
        },
        {
          "caption": "User picked [Propens...]",
          "condition": "temp['skill-choice-ret-p1jd9nzc9l'] == \"Propensity to
adverse reactions to substance\"",
          "node": "name-the-propensity-to-substance"
        },
        {
          "caption": "On failure",
          "condition": "true",
          "node": "error.flow.json"
        }
      ],
      "onEnter": null,
      "onReceive": null
    },
    {
      "id": "22c033c208",
      "name": "name-the-drug-allergies",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"allergy\",\"id\":\"{{session.userId}}\",\"category\":\"Drug
allergy\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "0b24e50f47",
      "name": "name-the-food-intolerance",
      "next": [
        {
```

```
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"intolerance\",\"id\":\"{{session.userId}}\",\"category\":\"Food
intolerance\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "f4d0ef788e",
      "name": "name-the-drug-intolerance",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"intolerance\",\"id\":\"{{session.userId}}\",\"category\":\"Drug
intolerance\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
```

```
    },
    {
      "id": "5866119286",
      "name": "name-the-substance-allergies",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"allergy\",\"id\":\"{{session.userId}}\",\"category\":\"Allergy to
substance\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "7912c68e71",
      "name": "name-the-propensity-to-food",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
```

```
        "get_allergies_info
{\"name\":\"propensity\",\"id\":\"{{session.userId}}\",\"category\":\"Propensity
to adverse reactions to food\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "c922d5cca1",
      "name": "name-the-propensity-to-substance",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"propensity\",\"id\":\"{{session.userId}}\",\"category\":\"Propensity
to adverse reactions to substance\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "5764212a3c",
      "name": "name-the-propensity-to-drug",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
```

```
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"propensity\",\"id\":\"{{session.userId}}\",\"category\":\"Propensity
to adverse reactions to drug\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "afccb5860a",
      "name": "name-the-intolerance",
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
          "node": "more-details"
        }
      ],
      "onEnter": [
        "get_allergies_info
{\"name\":\"intolerance\",\"id\":\"{{session.userId}}\",\"category\":\"Intolerance
\"}",
        "say #!builtin_text-kMLEkf"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "d14e074a60",
      "name": "no-allergies",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
```

```
        }
      ],
      "onEnter": [
        "say #!builtin_text-BB8xQC"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **help.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "c44092f85a",
      "name": "entry",
      "onEnter": [
        "say #!builtin_text-Tt3_bN",
        "say #!builtin_text-bJ4QbD"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "type": "standard"
    }
  ]
}
```

- **end_flow.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "23059de6cf",
      "name": "entry",
      "onEnter": [],
      "onReceive": null,
```

```
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "exit-confirmation"
        }
      ],
      "type": "standard"
    },
    {
      "id": "d4a82cf6a4",
      "name": "exiting-node",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "END"
        }
      ],
      "onEnter": [
        "say #!builtin_text-jy_0fc"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "skill-dbb52b",
      "type": "skill-call",
      "skill": "choice",
      "name": "exit-confirmation",
      "flow": "skills/choice-dbb52b.flow.json",
      "next": [
        {
          "caption": "User picked [yes]",
          "condition": "temp['skill-choice-ret-q9pvwv0qzy'] == \"yes\"",
          "node": "exiting-node"
        },
        {
          "caption": "User picked [no]",
          "condition": "temp['skill-choice-ret-q9pvwv0qzy'] == \"no\"",
          "conditionType": "raw",
          "node": "main.flow.json#questions_anwsers"
        },
        {
          "caption": "On failure",
          "condition": "true",
          "conditionType": "always",
          "node": "help.flow.json"
        }
      ],
```

```
      "onEnter": null,
      "onReceive": null
    }
  ]
}
```

- **error.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "nodes": [
    {
      "id": "3rr0r",
      "name": "entry",
      "onEnter": [
        "say #!builtin_text-Pb8akc"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "END"
        }
      ]
    }
  ]
}
```

- **medical_devices_implants.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "4d851f9d8c",
      "name": "medical-devices-info",
      "next": [
        {
          "condition": "true",
          "node": "implants-info"
        }
      ],
      "onEnter": [
        "get_medical_devices_info
{\"id\":\"{{session.userId}}\",\"description\":\"medical_devices\"}",
        "say #!builtin_text-fY3Lba"
```

```
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "b3f4d17c0e",
      "name": "implants-info",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "get_medical_devices_info
{\"id\":\"{{session.userId}}\",\"description\":\"implants\"}",
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "99e6676269",
      "name": "entry",
      "onEnter": [
        "exist-check
{\"table\":\"medical_devices_mapping\",\"key\":\"count(coding_id)\",\"whereColumn\
":\"patient_id\",\"value\":\"{{session.userId}}\",\"hash\":\"0\",\"database\":\"my
Database\"}"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "user.data == 0",
          "conditionType": "raw",
          "node": "no-devices-or-implants"
        },
        {
          "condition": "true",
          "node": "found"
        }
      ],
      "type": "standard"
    },
    {
      "id": "185a23d9f9",
      "name": "no-devices-or-implants",
      "next": [
        {
```

```
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "say #!builtin_text-j7UeBc"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "316d8491ba",
      "name": "found",
      "next": [
        {
          "condition": "true",
          "node": "medical-devices-info"
        }
      ],
      "onEnter": [
        "say #!builtin_text-5tSajm"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **past_illnesses.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "e9f508379c",
      "name": "entry",
      "onEnter": [
        "exist-check
{\"table\":\"illness_mapping\",\"key\":\"count(coding_id)\",\"whereColumn\":\"pati
ent_id\",\"value\":\"{{session.userId}}\",\"hash\":\"0\",\"database\":\"myDatabase
\"}"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "user.data == 0",
          "conditionType": "raw",
```

```
        "node": "no-past-illnesses"
      },
      {
        "condition": "true",
        "node": "found"
      }
    ],
    "type": "standard"
  },
  {
    "id": "skill-fe0257",
    "type": "skill-call",
    "skill": "choice",
    "name": "find-more-about-illness",
    "flow": "skills/choice-fe0257.flow.json",
    "next": [
      {
        "caption": "User picked [yes]",
        "condition": "temp['skill-choice-ret-ti6p5z6nub'] == \"yes\"",
        "node": "past-illnesses-info"
      },
      {
        "caption": "User picked [no]",
        "condition": "temp['skill-choice-ret-ti6p5z6nub'] == \"no\"",
        "conditionType": "raw",
        "node": "main.flow.json#questions_anwsers"
      },
      {
        "caption": "On failure",
        "condition": "true",
        "conditionType": "always",
        "node": "error.flow.json"
      }
    ],
    "onEnter": null,
    "onReceive": null
  },
  {
    "id": "53a2046164",
    "name": "past-illnesses-info",
    "next": [
      {
        "condition": "event.nlu.intent.name === 'log_out'",
        "conditionType": "intent",
        "node": "end_flow.flow.json"
      },
      {
        "condition": "event.nlu.intent.name === 'help'",
        "conditionType": "intent",
        "node": "help.flow.json"
```

```
        },
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "get_illnesses_info {\"id\":\"{{session.userId}}\",\"ongoing\":\"0\"}",
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "4b7723f8ea",
      "name": "found",
      "next": [
        {
          "condition": "true",
          "node": "find-more-about-illness"
        }
      ],
      "onEnter": [
        "say #!builtin_text-wC43rQ"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "40eeecbc85",
      "name": "no-past-illnesses",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "say #!builtin_text-dwZz6Z"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **current_problems.flow.json**


```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "70b59a1e8e",
      "name": "entry",
      "onEnter": [],
      "onReceive": null,
      "next": [
        {
          "condition": "true",
          "node": "which-problem"
        }
      ],
      "type": "standard"
    },
    {
      "id": "skill-3ad6ba",
      "type": "skill-call",
      "skill": "choice",
      "name": "which-problem",
      "flow": "skills/choice-3ad6ba.flow.json",
      "next": [
        {
          "caption": "User picked [Ongoing...]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"Ongoing
illnesses\"",
          "node": "ongoing-illnesses"
        },
        {
          "caption": "User picked [Allergi...]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"Allergies\"",
          "node": "allergies.flow.json"
        },
        {
          "caption": "User picked [Illness...]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"Illnesses I've
had in the last  6 months\"",
          "node": "last-6-months-illnesses"
        },
        {
          "caption": "User picked [social ...]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"social
problems\"",
          "node": "Social-Problems"
        },
```

```
        {
          "caption": "User picked [Somethi...]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"Something
else\"",
          "conditionType": "raw",
          "node": "main.flow.json#ask-questions"
        },
        {
          "caption": "User picked [Nothing]",
          "condition": "temp['skill-choice-ret-h1ifkdr1l7'] == \"Nothing\"",
          "node": "end_flow.flow.json"
        },
        {
          "caption": "On failure",
          "condition": "true",
          "node": "error.flow.json"
        }
      ],
      "onEnter": null,
      "onReceive": null
    },
    {
      "id": "15f2f277af",
      "name": "ongoing-illnesses",
      "next": [
        {
          "condition": "true",
          "node": "which-problem"
        }
      ],
      "onEnter": [
        "get_illnesses_info {\"id\":\"{{session.userId}}\",\"ongoing\":\"1\"}",
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "4e2c6451bd",
      "name": "last-6-months-illnesses",
      "next": [
        {
          "condition": "true",
          "node": "which-problem"
        }
      ],
      "onEnter": [
        "get_illnesses_info {\"id\":\"{{session.userId}}\",\"ongoing\":\"6\"}",
        "say #!builtin_text-fY3Lba"
      ],
```

```
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "03f45b48e3",
      "name": "Social-Problems",
      "next": [
        {
          "condition": "true",
          "node": "which-problem"
        }
      ],
      "onEnter": [
        "get_social_history_info {\"id\":\"{{session.userId}}\"}",
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **prescription_summary.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "7042cfbd37",
      "name": "entry",
      "onEnter": [
        "get_perscription_info {\"id\":\"{{session.userId}}\"}"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "true",
          "node": "present-prescription"
        }
      ],
      "type": "standard"
    },
    {
      "id": "b557d32714",
      "name": "present-prescription",
      "next": [
        {
          "condition": "true",
```

```
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **surgeries.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "e60ac903d4",
      "name": "entry",
      "onEnter": [
        "exist-check
{\"table\":\"surgical_procedures_mapping\",\"key\":\"count(coding_id)\",\"whereCol
umn\":\"patient_id\",\"value\":\"{{session.userId}}\",\"hash\":\"0\",\"database\":
\"myDatabase\"}"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "user.data == 0",
          "conditionType": "raw",
          "node": "no-surgeries"
        },
        {
          "condition": "true",
          "node": "found"
        }
      ],
      "type": "standard"
    },
    {
      "id": "a84a61b186",
      "name": "surgeries-info",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
```

```
        "node": "main.flow.json#questions_anwsers"
      }
    ],
    "onEnter": [
      "get_surgeries_info {\"id\":\"{{session.userId}}\"}",
      "say #!builtin_text-fY3Lba"
    ],
    "onReceive": null,
    "type": "standard"
  },
  {
    "id": "skill-1b8f5d",
    "type": "skill-call",
    "skill": "choice",
    "name": "find-more-about-surgeries",
    "flow": "skills/choice-1b8f5d.flow.json",
    "next": [
      {
        "caption": "User picked [yes]",
        "condition": "temp['skill-choice-ret-P5xJckFHfg'] == \"yes\"",
        "node": "surgeries-info"
      },
      {
        "caption": "User picked [no]",
        "condition": "temp['skill-choice-ret-P5xJckFHfg'] == \"no\"",
        "conditionType": "raw",
        "node": "main.flow.json#questions_anwsers"
      },
      {
        "caption": "On failure",
        "condition": "true",
        "conditionType": "always",
        "node": "error.flow.json#questions_anwsers"
      }
    ],
    "onEnter": null,
    "onReceive": null
  },
  {
    "id": "05631f6435",
    "name": "found",
    "next": [
      {
        "condition": "true",
        "node": "find-more-about-surgeries"
      }
    ],
    "onEnter": [
      "say #!builtin_text-Kgfzv9"
    ],
```

```
      "onReceive": null,
      "type": "standard"
    },
    {
      "id": "7faa69da22",
      "name": "no-surgeries",
      "next": [
        {
          "condition": "true",
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "onEnter": [
        "say #!builtin_text-HD62_V"
      ],
      "onReceive": null,
      "type": "standard"
    }
  ]
}
```

- **vaccinations.flow.json**

```
{
  "version": "0.0.1",
  "catchAll": {},
  "startNode": "entry",
  "description": "",
  "nodes": [
    {
      "id": "06b458ab49",
      "name": "entry",
      "onEnter": [
        "get_vaccinations_info {\"id\":\"{{session.userId}}\"}",
        "say #!builtin_text-fY3Lba"
      ],
      "onReceive": null,
      "next": [
        {
          "condition": "event.nlu.intent.name === 'help'",
          "conditionType": "intent",
          "node": "help.flow.json"
        },
        {
          "condition": "event.nlu.intent.name === 'log_out'",
          "conditionType": "intent",
          "node": "end_flow.flow.json"
        },
        {
          "condition": "true",
```

```
          "conditionType": "always",
          "node": "main.flow.json#questions_anwsers"
        }
      ],
      "type": "standard"
    }
  ]
}
```

Παράρτημα Γ: Custom Actions

- **AES_encryption.js**

```
/**
 * Small description of your action
 * @title AES encryption - hash value
 * @category Utility
 * @param {string} id - the patients id
 * @param {string} date - the patients date of birth
 */
const myAction = async (id, date) => {
  const crypto = require('crypto')
  user.hash = undefined
  let secretKey
  let key
  let myKey = ´secret key´
  let encrypted

  try {
    /*set key*/
    try {
      key = Buffer.from(myKey, 'utf8')
      const sha = crypto.createHash('sha1')
      key = sha.update(key).digest()
      key = key.slice(0, 16)
      secretKey = crypto.createCipheriv('aes-128-ecb', key, '')
    } catch (e) {
      console.error(`Error while setting key: ${e}`)
    }
    bp.logger.info('Secret key is : ' + secretKey)

    /*set the string you want to encrypt*/
    let strToEncrypt = date + id
    encrypted = secretKey.update(strToEncrypt, 'utf8', 'base64')
    encrypted += secretKey.final('base64')
    // Reading data
    crypto.on('readable', () => {
      let chunk
      while (null !== (chunk = crypto.read())) {
        encrypted += chunk.toString('base64')
      }
    })
  } catch (e) {
    console.error(`Error while decrypting: ${e}`)
  }

  bp.logger.info('Hash: ' + encrypted)
  // Encode the string to Base64
  const encodedString = Buffer.from(encrypted, 'utf-8').toString('base64')
```

```
    user.hash = encrypted
    bp.logger.info(user.hash)
  }

  return myAction(args.id, args.date)
```

- **exist-check.js**

```
  /**
   * This utility can check if the key exist in MySQL DB
   * @title User MySQL DB Utility
   * @category Utility
   * @param {string} table - Table name
   * @param {string} key - The column we want to check if exist
   * @param {string} whereColumn - The column in the where clause
   * @param {string} value - Can contain a any value which is the value to check
in where clause
   * @param {string} database - The name of the target database
   * @param {string} hash - If it is 1 then we have a hash value, if it is 0 then
we have normal string
   */
  const userDBUtility = async (table, key, whereColumn, value, database, hash) =>
{
    if (hash === '1') {
      bp.logger.info(value)
      const temp = value.replace(/&#x2F;/g, '/').replace(/&#x3D;/g, '=')
      value = temp
      bp.logger.info(value)
    }

    bp.logger.info(table + ' ' + key + ' ' + whereColumn + ' ' + value)
    user.data = undefined
    user.response = undefined
    const knex = require('knex')({
      client: 'mysql',
      connection: {
        host: 'botpress.ucycodehub.eu',
        port: 30340,
        user: 'root',
        password: 'password',
        database: database
      },
      useNullAsDefault: false,
      log: {
        warn(message) {
          console.log(message)
        },
        error(message) {
          console.error(message)
```

```
        },
        deprecate(message) {
          console.log(message)
        },
        debug(message) {
          console.log(message)
        }
      }
    })

    if (knex) {
      temp.param = value

      let query = ''
      if (hash === '1') {
        query = 'select ' + key + ' from ' + table + ' where ' + whereColumn + '=
md5("' + value + '")'
      } else {
        query = 'select ' + key + ' from ' + table + ' where ' + whereColumn + "
='" + value + "'"
      }
      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place. Useful for
logging all queries throughout your application.
        bp.logger.info('Executing: ' + data.sql)
      })
      if (data[0][0] === undefined) {
        user.data = undefined
        user.response = 'no'
      } else {
        user.data = data[0][0][key]
        user.response = 'yes'
      }
    } else {
      bp.logger.info('knex is not initialized')
    }
  }

  return userDBUtility(args.table, args.key, args.whereColumn, args.value,
args.database, args.hash)
```

- **get_allergies_info.js**

```
/**
 * This utility can Get data from MySQL DB
 * @title Allergies_info
 * @category Utility
 * @param {string} name - can be allergy, intolerance, propensity
 * @param {string} id - the patient id
 * @param {string} category - the description of the allergies
 */
const userDBUtility = async (name, id, category) => {
  user.data = undefined
  //established the connection
  const knex = require('knex')({
    client: 'mysql',
    connection: {
      host: 'botpress.ucycodehub.eu',
      port: 30340,
      user: 'root',
      password: 'password',
      database: 'myDatabase'
    },
    useNullAsDefault: false,
    log: {
      warn(message) {
        console.log(message)
      },
      error(message) {
        console.error(message)
      },
      deprecate(message) {
        console.log(message)
      },
      debug(message) {
        console.log(message)
      }
    }
  })

  if (knex) {
    let query = ''
    //create the query
    if (name === 'allergy' || name === 'intolerance' || name === 'propensity') {
      query =
        "SELECT c0.description as allergy_desc, c1.description as reaction_desc,
c2.description AS agent_desc FROM `allergies_mapping` AS a INNER JOIN codings as
c0 ON c0.coding_id = a.allergy_coding_id INNER JOIN codings as c1 ON c1.coding_id
= a.reaction_coding_id INNER JOIN codings as c2 ON c2.coding_id =
a.agent_coding_id INNER JOIN ps_versioning  as v ON v.id = a.version_id INNER JOIN
u_users  as u ON u.user_id = v.user_id WHERE a.patient_id ='" +
```

```
          id +
          "' and v.id <= (SELECT IF(30 > -1, 30, max(v.id))) and c0.description =
'" +
          category +
          "'"
        bp.logger.info(query)
      }

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined && name === 'allergy') {
        user.data = ''
        user.data += 'You do not have any allergies in this category!'
      } else if (data[0][0] === undefined && name === 'intolerance') {
        user.data = ''
        user.data += 'You do not have any intolerances in this category!'
      } else if (data[0][0] === undefined && name === 'propensity') {
        user.data = ''
        user.data += 'You do not have any propensity to adverse reactions in this
category!'
      } else if (name === 'allergy') {
        user.data = ''
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i + 1 + ') Allergy to ' + data[0][i]['agent_desc'] + ' (Reaction: ' +
data[0][i]['reaction_desc'] + ')\n \n'
        }
      } else if (name === 'intolerance') {
        user.data = ''
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i +
            1 +
            ') Intolerance to ' +
            data[0][i]['agent_desc'] +
            ' (Reaction: ' +
            data[0][i]['reaction_desc'] +
            ')\n \n'
        }
      } else if (name === 'propensity') {
        user.data = ''
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i +
            1 +
            ') Propensity to adverse reaction to ' +
```

```
            data[0][i]['agent_desc'] +
            ' (Reaction: ' +
            data[0][i]['reaction_desc'] +
            ')\n \n'
        }
      }
    } else {
      bp.logger.info('knex is not initialized')
    }
  }

  return userDBUtility(args.name, args.id, args.category)
```

- **get_illnesses_info.js**

```
  /**
   * This utility can Get data from MySQL DB
   * @title Illnesses_info
   * @category Utility
   * @param {string} id - the patient id
   * @param {string} ongoing - it can be 1(ongoing illnesses) or 0(all illnesses)
or 6(illness tha last 6 months)
   */
  const myAction = async (id, ongoing) => {
    user.data = undefined
    //established the connection
    const knex = require('knex')({
      client: 'mysql',
      connection: {
        host: 'botpress.ucycodehub.eu',
        port: 30340,
        user: 'root',
        password: 'password',
        database: 'myDatabase'
      },
      useNullAsDefault: false,
      log: {
        warn(message) {
          console.log(message)
        },
        error(message) {
          console.error(message)
        },
        deprecate(message) {
          console.log(message)
        },
        debug(message) {
          console.log(message)
        }
```

```
      }
    })

    if (knex) {
      let query = ''
      //create the query
      if (ongoing === '0') {
        query =
          "SELECT c0.description as
illness_desc,ill_map.resolution_circumstances,DATE_FORMAT(ill_map.onset_date, '%Y-
%m-%d') as start_date,DATE_FORMAT(ill_map.end_date, '%Y-%m-%d') as end_date FROM
`illness_mapping` AS ill_map INNER JOIN codings as c0 ON c0.coding_id =
ill_map.coding_id INNER JOIN ps_versioning  as ver ON ver.id = ill_map.version_id
INNER JOIN u_users  as u ON  u.user_id = ver.user_id where ill_map.patient_id = '"
+
          id +
          "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) order by
start_date DESC"
      } else if (ongoing === '1') {
        query =
          "SELECT c0.description as
illness_desc,ill_map.resolution_circumstances,DATE_FORMAT(ill_map.onset_date, '%Y-
%m-%d') as start_date,DATE_FORMAT(ill_map.end_date, '%Y-%m-%d') as end_date FROM
`illness_mapping` AS ill_map INNER JOIN codings as c0 ON c0.coding_id =
ill_map.coding_id INNER JOIN ps_versioning  as ver ON ver.id = ill_map.version_id
INNER JOIN u_users  as u ON  u.user_id = ver.user_id where ill_map.patient_id = '"
+
          id +
          "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) and
ill_map.end_date is NULL order by start_date DESC"
      } else if (ongoing === '6') {
        query =
          "SELECT c0.description as
illness_desc,ill_map.resolution_circumstances,DATE_FORMAT(ill_map.onset_date, '%Y-
%m-%d') as start_date,DATE_FORMAT(ill_map.end_date, '%Y-%m-%d') as end_date FROM
`illness_mapping` AS ill_map INNER JOIN codings as c0 ON c0.coding_id =
ill_map.coding_id INNER JOIN ps_versioning  as ver ON ver.id = ill_map.version_id
INNER JOIN u_users  as u ON  u.user_id = ver.user_id where ill_map.patient_id = '"
+
          id +
          "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) and
ill_map.end_date > DATE_SUB(now(), INTERVAL 6 MONTH) order by start_date DESC"
      }
      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })
```

```
    if (data[0][0] === undefined) {
      user.data = ''
      if (ongoing === '6') {
        user.data += 'You have not had any illness the last 6 months.'
      } else if (ongoing === '1') {
        user.data += 'You do not have any ongoing illness.'
      } else {
        user.data += 'You do not have any past illness in your history.'
      }
    } else if (data[0].length != 0) {
      user.data = ''
      for (let i = 0; i < data[0].length; i++) {
        let rc = ''
        if (data[0][i]['resolution_circumstances'] != null) {
          rc = data[0][i]['resolution_circumstances']
        } else {
          rc = 'nothing'
        }
        let end_d = ''
        if (data[0][i]['end_date'] != null) {
          end_d += ' to ' + data[0][i]['end_date']
        } else {
          end_d = ' to ongoing'
        }
        user.data +=
          i +
          1 +
          ') ' +
          data[0][i]['illness_desc'] +
          '\n (Duration: ' +
          data[0][i]['start_date'] +
          end_d +
          ')\n Resolution Circumstances: ' +
          rc +
          '\n \n'
      }
    } else {
      bp.logger.info('knex is not initialized')
    }
  }
}

return myAction(args.id, args.ongoing)
```

- **get_medical_devices_info.js**

```
/**
 * Small description of your action
 * @title Medical Devices Info
 * @category Custom
 * @param {string} id - the patient id
 * @param {string} description - it can be implants or medical_devices
 */
const myAction = async (id, description) => {
  user.data = undefined
  //established the connection
  const knex = require('knex')({
    client: 'mysql',
    connection: {
      host: 'botpress.ucycodehub.eu',
      port: 30340,
      user: 'root',
      password: 'password',
      database: 'myDatabase'
    },
    useNullAsDefault: false,
    log: {
      warn(message) {
        console.log(message)
      },
      error(message) {
        console.error(message)
      },
      deprecate(message) {
        console.log(message)
      },
      debug(message) {
        console.log(message)
      }
    }
  })

  if (knex) {
    let query = ''
    //create the query
    if (description === 'implants') {
      query =
        "SELECT c0.description as md_descrtiption,DATE_FORMAT(md.implant_date,
'%Y-%m-%d') as implant_date FROM `medical_devices_mapping` AS md INNER JOIN
codings as c0 ON c0.coding_id = md.coding_id INNER JOIN ps_versioning  as ver ON
ver.id = md.version_id INNER JOIN u_users  as u ON  u.user_id = ver.user_id where
md.patient_id = '" +
        id +
```

```
          "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) and
c0.description LIKE '%implant%' order by implant_date DESC"
      } else if (description === 'medical_devices') {
        query =
          "SELECT c0.description as md_descrtiption,DATE_FORMAT(md.implant_date,
'%Y-%m-%d') as implant_date FROM `medical_devices_mapping` AS md INNER JOIN
codings as c0 ON c0.coding_id = md.coding_id INNER JOIN ps_versioning  as ver ON
ver.id = md.version_id INNER JOIN u_users  as u ON  u.user_id = ver.user_id where
md.patient_id = '" +
          id +
          "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) and
c0.description NOT LIKE '%implant%' order by implant_date DESC"
      }

      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined) {
        user.data = ''
        if (description === 'implants') {
          user.data += 'No implants.'
        } else {
          user.data += 'No medical devices.'
        }
      } else if (data[0].length != 0) {
        user.data = ''
        if (description === 'implants') {
          user.data += 'Implants: \n \n'
        } else {
          user.data += 'Medical Devices: \n \n'
        }
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i + 1 + ') ' + data[0][i]['md_descrtiption'] + '\n (Date: ' +
data[0][i]['implant_date'] + ')\n \n'
        }
      } else {
        bp.logger.info('knex is not initialized')
      }
    }
  }

  return myAction(args.id, args.description)
```

- **get_prescription_info.js**

```
/**
 * Small description of your action
 * @title perscription info
 * @category Custom
 * @param {string} id -  the patient id
 */
const myAction = async id => {
  user.data = undefined
  //established the connection
  const knex = require('knex')({
    client: 'mysql',
    connection: {
      host: 'botpress.ucycodehub.eu',
      port: 30340,
      user: 'root',
      password: 'password',
      database: 'myDatabase'
    },
    useNullAsDefault: false,
    log: {
      warn(message) {
        console.log(message)
      },
      error(message) {
        console.error(message)
      },
      deprecate(message) {
        console.log(message)
      },
      debug(message) {
        console.log(message)
      }
    }
  })

  if (knex) {
    let query = ''
    //create the query
    query =
      "SELECT c0.description as active_ingredient,c1.description as
pharmaceutical_form,pm.brand_name,pm.number_of_units_per_intake as
dose,c2.description as unit_id,pm.frequency_of_intake as frequency,c3.description
as frequency_id,pm.duration as duration,c4.description as
duration_id,pm.instructions_to_patient as
instructions,DATE_FORMAT(pm.date_of_issue, '%Y-%m-%d') as
date_of_issue,c5.description as medicine FROM `prescription_mapping` AS pm INNER
JOIN codings as c0 ON c0.coding_id = pm.active_ingredient_id INNER JOIN codings as
c1 ON c1.coding_id = pm.pharmaceutical_dose_form_id INNER JOIN codings as c2 ON
```

```
c2.coding_id = pm.unit_id INNER JOIN codings as c3 ON c3.coding_id =
pm.frequency_of_intake_id INNER JOIN codings as c4 ON c4.coding_id =
pm.duration_unit_id INNER JOIN ps_versioning  as ver ON ver.id = pm.version_id
INNER JOIN u_users  as u ON u.user_id = ver.user_id INNER JOIN codings as c5 on
pm.prescription_mapping_id=c5.coding_id where pm.patient_id = '" +
        id +
        "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) order by
date_of_issue DESC"

      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined) {
        user.data = ''
        user.data += 'No perscription.'
      } else if (data[0].length != 0) {
        user.data = 'You have the following prescriptions in your history: \n'
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i +
            1 +
            ') ' +
            data[0][i]['medicine'] +
            '\n Active Ingredient: ' +
            data[0][i]['active_ingredient'] +
            '\n Form: ' +
            data[0][i]['pharmaceutical_form'] +
            '\n Brand: ' +
            data[0][i]['brand_name'] +
            '\n Unit dose: ' +
            data[0][i]['dose'] +
            //' ' +
            //data[0][i]['unit_id'] +
            '\n Frequency: ' +
            data[0][i]['frequency'] +
            ' ' +
            data[0][i]['frequency_id'] +
            '\n Duration: ' +
            data[0][i]['duration'] +
            ' ' +
            data[0][i]['duration_id'] +
            '\n Instructions: ' +
            data[0][i]['instructions'] +
            '\n (Date of issue: ' +
            data[0][i]['date_of_issue'] +
```

```
          ')\n \n'
      }
    } else {
      bp.logger.info('knex is not initialized')
    }
  }
}

return myAction(args.id)
```

- **get_social_history_info.js**

```
/**
 * Small description of your action
 * @title Social History
 * @category Custom
 * @param {string} id - the patient id
 */
const myAction = async id => {
  user.data = undefined
  //established the connection
  const knex = require('knex')({
    client: 'mysql',
    connection: {
      host: 'botpress.ucycodehub.eu',
      port: 30340,
      user: 'root',
      password: 'password',
      database: 'myDatabase'
    },
    useNullAsDefault: false,
    log: {
      warn(message) {
        console.log(message)
      },
      error(message) {
        console.error(message)
      },
      deprecate(message) {
        console.log(message)
      },
      debug(message) {
        console.log(message)
      }
    }
  })

  if (knex) {
    let query = ''
```

```
      //create the query
      query =
        "SELECT c0.description as description,DATE_FORMAT(sh.start_date, '%Y-%m-
%d') as start_date,DATE_FORMAT(sh.end_date, '%Y-%m-%d') as end_date FROM
`social_history_mapping` AS sh INNER JOIN codings as c0 ON c0.coding_id =
sh.coding_id INNER JOIN ps_versioning  as ver ON ver.id = sh.version_id INNER JOIN
u_users  as u ON u.user_id = ver.user_id where sh.patient_id = '" +
        id +
        "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) order by start_date
DESC"

      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined) {
        user.data = ''
        user.data += 'No Social History.'
      } else if (data[0].length != 0) {
        user.data = ''
        for (let i = 0; i < data[0].length; i++) {
          user.data +=
            i +
            1 +
            ') ' +
            data[0][i]['description'] +
            '\n (Duration: ' +
            data[0][i]['start_date'] +
            ' - ' +
            data[0][i]['end_date'] +
            ')\n \n'
        }
      } else {
        bp.logger.info('knex is not initialized')
      }
    }
  }

  return myAction(args.id)
```

- **get_surgeries_info.js**

```
/**
 * This utility can Get data from MySQL DB
 * @title Surgeries_info
 * @category Utility
 * @param {string} id - the patient id
 */
const myAction = async id => {
  user.data = undefined
  //established the connection
  const knex = require('knex')({
    client: 'mysql',
    connection: {
      host: 'botpress.ucycodehub.eu',
      port: 30340,
      user: 'root',
      password: 'password',
      database: 'myDatabase'
    },
    useNullAsDefault: false,
    log: {
      warn(message) {
        console.log(message)
      },
      error(message) {
        console.error(message)
      },
      deprecate(message) {
        console.log(message)
      },
      debug(message) {
        console.log(message)
      }
    }
  })

  if (knex) {
    let query = ''
    //create the query
    query =
      "SELECT c0.description as surgery_desc,DATE_FORMAT(sp.procedure_date, '%Y-
%m-%d') as sp_date FROM `surgical_procedures_mapping` AS sp INNER JOIN codings as
c0 ON c0.coding_id = sp.coding_id INNER JOIN ps_versioning  as ver ON ver.id =
sp.version_id INNER JOIN u_users  as u ON  u.user_id = ver.user_id where
sp.patient_id = '" +
      id +
      "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) order by sp_date
DESC"
```

```
      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined) {
        user.data = ''
        user.data += 'No surgical procedures in your history.'
      } else if (data[0].length != 0) {
        user.data = ''
        for (let i = 0; i < data[0].length; i++) {
          user.data += i + 1 + ') ' + data[0][i]['surgery_desc'] + '\n (Date: ' +
data[0][i]['sp_date'] + ')\n \n'
        }
      } else {
        bp.logger.info('knex is not initialized')
      }
    }
  }

  return myAction(args.id)
```

- **get_vaccinations_info.js**

```
  /**
   * This utility can Get data from MySQL DB
   * @title Vaccinations_info
   * @category Utility
   * @param {string} id - the patient id
   */
  const myAction = async id => {
    user.data = undefined
    //established the connection
    const knex = require('knex')({
      client: 'mysql',
      connection: {
        host: 'botpress.ucycodehub.eu',
        port: 30340,
        user: 'root',
        password: 'password',
        database: 'myDatabase'
      },
      useNullAsDefault: false,
      log: {
        warn(message) {
          console.log(message)
        },
        error(message) {
```

```
          console.error(message)
        },
        deprecate(message) {
          console.log(message)
        },
        debug(message) {
          console.log(message)
        }
      }
    })

    if (knex) {
      let query = ''
      //create the query
      query =
        "SELECT c0.description as vaccination_desc,
DATE_FORMAT(v.vaccination_date, '%Y-%m-%d') as vac_date FROM
`vaccinations_mapping` AS v INNER JOIN codings as c0 ON c0.coding_id = v.coding_id
INNER JOIN ps_versioning  as ver ON ver.id = v.version_id INNER JOIN u_users  as u
ON u.user_id = ver.user_id where v.patient_id = '" +
        id +
        "' and ver.id <= (SELECT IF(30 > -1, 30, max(ver.id))) order by vac_date
DESC"

      bp.logger.info(query)

      //wait for response
      const data = await knex.raw(query).on('query', function(data) {
        //A query event is fired just before a query takes place.
        console.log('Executing: ' + data.sql)
      })

      if (data[0][0] === undefined) {
        user.data = ''
        user.data += 'No vaccinations.'
      } else if (data[0].length != 0) {
        user.data = ''
        user.data += 'You have done the following vaccines: \n \n'
        for (let i = 0; i < data[0].length; i++) {
          user.data += i + 1 + ') ' + data[0][i]['vaccination_desc'] + '\n (Date:
' + data[0][i]['vac_date'] + ')\n \n'
        }
      } else {
        bp.logger.info('knex is not initialized')
      }
    }
  }

  return myAction(args.id)
```