ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	BookLovers Website: Ανάπτυξη μιας διαδικτυακής εφαρμογής προτάσεων βιβλίων βασισμένη στη μηχανική μάθηση BookLovers Website: Development of a Web Application for Book Recommendations Based on Machine Learning
Ονοματεπώνυμο Φοιτητή	ΣΚΛΗΡΗΣ ΠΑΝΑΓΙΩΤΗΣ
Πατρώνυμο	Βασιλειος
Αριθμός Μητρώου	Π/ 20175
Επιβλέπων	Σακκόπουλος Ευάγγελος, Καθηγητής

Ημερομηνία Παράδοσης Ιούνιος 2026

Copyright ©

Επιτελική Σύνοψη

Η παρούσα επιστημονική εργασία πραγματεύεται την υλοποίηση μιας διαδικτυακής εφαρμογής προτάσεων βιβλίων βασισμένης σε τεχνικές μηχανικής μάθησης. Η δημιουργία της κρίθηκε αναγκαία λόγω της ραγδαίας αύξησης του διαθέσιμου ψηφιακού βιβλιογραφικού περιεχομένου, η οποία καθιστά ιδιαίτερα δύσκολο για τον χρήστη τον εντοπισμό βιβλίων που ανταποκρίνονται στις προτιμήσεις και τα ενδιαφέροντά του. Τα προβλήματα αυτά χρήζουν επίλυσης εφόσον, αν και υφίστανται ορισμένες διαδικτυακές πλατφόρμες που παρέχουν αντίστοιχες υπηρεσίες, οι δυνατότητες που αυτές προσφέρουν δεν επαρκούν για την αποτελεσματική εξατομίκευση των προτάσεων με βάση τις ιδιαίτερες ανάγκες κάθε χρήστη. Τα ζητήματα αυτά επιλύονται μέσω της ανάπτυξης της συγκεκριμένης εφαρμογής, η οποία αποτελεί και τον κεντρικό άξονα της δεδομένης εργασίας. Η υλοποίησή της πραγματοποιήθηκε με τη χρήση του μικροπλαισίου Flask της γλώσσας προγραμματισμού Python, ενώ για τη διαχείριση των δεδομένων επιλέχθηκε το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων SQLite. Τα συγκεκριμένα εργαλεία παρέχουν πληθώρα δυνατοτήτων, οι οποίες αξιοποιήθηκαν για την υλοποίηση των λειτουργιών που προσφέρει το εν λόγω σύστημα. Στην ευκολότερη κατανόηση και απλοποίηση των απαιτήσεων που σχετίζονται με τη δημιουργία του κατείχαν κυρίαρχο ρόλο τόσο η σχεδίαση κατάλληλων διαγραμμάτων αναφορικά με την αρχιτεκτονική της εφαρμογής όσο και ο ορισμός των οντοτήτων της βάσης δεδομένων και των μεταξύ τους σχέσεων. Τελικά, η διαδικτυακή αυτή πλατφόρμα, η οποία θα αναλυθεί εκτενώς στο παρόν έγγραφο, αξιοποιεί αλγορίθμους μηχανικής μάθησης για την παροχή εξατομικευμένων προτάσεων βιβλίων, προσφέροντας ποικίλες λειτουργίες σε χρήστες που αφορούν τόσο τη διαχείριση του προφίλ τους όσο και την ανακάλυψη νέων τίτλων βιβλίων βάσει των αναγνωστικών τους προτιμήσεων.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επίκουρο και επιβλέποντα καθηγητή μου στα πλαίσια εκπόνησης της πτυχιακής μου εργασίας κ. Ευάγγελο Σακκόπουλο για τη δυνατότητα που μου έδωσε να πραγματοποιήσω την διατριβή μου, και για την υποστήριξη που μου παρείχε καθώς και την εμπιστοσύνη που μου έδειξε κατά την υλοποίησή της.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Πανεπιστημίου Πειραιώς για την πληθώρα των πολύτιμων γνώσεων που μου προσέφεραν πάνω στην πληροφορική όλα αυτά τα χρόνια της φοίτησής μου καθώς και τους γονείς μου, οι οποίοι με την αμέριστη συμπαράστασή τους, την εμπιστοσύνη που έδειξαν στο πρόσωπό μου και την ενθάρρυνσή τους καθ' όλη τη διάρκεια των σπουδών μου, αποτέλεσαν την κινητήριο δύναμη για την ολοκλήρωση της παρούσας εργασίας.

Ιούνιος 2026
Σκλήρης Παναγιώτης

1.1.1 ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΌΝΩΝ

Κεφάλαιο 1ο

1 Εισαγωγή

1.1 Περιγραφή του υπό μελέτη προβλήματος

Στη σύγχρονη ψηφιακή εποχή, η πρόσβαση στο διαδίκτυο έχει μεταβάλει ριζικά τις καταναλωτικές συνήθειες και τον τρόπο με τον οποίο οι άνθρωποι αναζητούν και επιλέγουν περιεχόμενο. Η ραγδαία εξάπλωση των ψηφιακών μέσων, των πλατφορμών κοινωνικής δικτύωσης και της διαδικτυακής ψυχαγωγίας έχει οδηγήσει σε σταδιακή μείωση του χρόνου που αφιερώνει το ευρύ κοινό στην ανάγνωση βιβλίων. Πολλοί άνθρωποι, ιδιαίτερα οι νεότερες γενιές, στρέφονται όλο και περισσότερο σε σύντομες και εύπεπτες μορφές περιεχομένου, όπως βίντεο, podcasts και αναρτήσεις στα μέσα κοινωνικής δικτύωσης, εις βάρος της ενασχόλησης με το παραδοσιακό βιβλίο.
Το φαινόμενο αυτό δεν αφορά μόνο την έντυπη μορφή του βιβλίου καθώς ακόμα και στον χώρο του ψηφιακού βιβλίου, η αποσπασματική προσοχή που χαρακτηρίζει τον σύγχρονο χρήστη του διαδικτύου καθιστά δυσκολότερη την ενασχόλησή του με εκτενή αναγνωστικά κείμενα. Ως αποτέλεσμα, ακόμα και όσοι διατηρούν αναγνωστικές συνήθειες, αντιμετωπίζουν συχνά το πρόβλημα της επιλογής· μπροστά σε εκατομμύρια διαθέσιμους τίτλους, το ερώτημα «ποιο βιβλίο να διαβάσω τώρα;» αποτελεί μια καθημερινή πρόκληση που συχνά οδηγεί σε αναποφασιστικότητα ή ακόμα και στην εγκατάλειψη της αναζήτησης.
Προς αντιμετώπιση του παραπάνω προβλήματος αναπτύχθηκε η διαδικτυακή εφαρμογή **BookLovers**, η οποία αξιοποιεί τεχνικές μηχανικής μάθησης με σκοπό την παροχή εξατομικευμένων προτάσεων βιβλίων. Στόχος της εφαρμογής είναι διττός: αφενός να διευκολύνει τον ήδη ενεργό αναγνώστη στην εύρεση του επόμενου βιβλίου του, και αφετέρου να λειτουργήσει ως κίνητρο επαναπροσέγγισης της ανάγνωσης για εκείνους που έχουν απομακρυνθεί από αυτήν. Μέσω στοχευμένων προτάσεων βασισμένων στις προτιμήσεις κάθε χρήστη, η εφαρμογή επιδιώκει να μειώσει τον χρόνο αναζήτησης, να ενισχύσει την αναγνωστική εμπειρία και να συμβάλει στην καλλιέργεια της αναγνωστικής κουλτούρας στο ευρύτερο κοινό.

1.2 Χρήση παρόμοιων λογισμικών

Η διαδικτυακή εφαρμογή BookLovers δεν είναι μοναδική ως προς την ιδέα παροχής προτάσεων βιβλίων μέσω διαδικτύου. Υπάρχουν ήδη ορισμένες πλατφόρμες που επιχειρούν να απαντήσουν στο ερώτημα της επιλογής βιβλίου, καθεμία με διαφορετική προσέγγιση και δυνατότητες. Το StoryGraph είναι μια σύγχρονη διαδικτυακή πλατφόρμα που επιτρέπει στους χρήστες να καταγράφουν τα βιβλία που έχουν διαβάσει, να τα αξιολογούν και να παρακολουθούν την αναγνωστική τους πρόοδο. Όταν ο χρήστης εισέρχεται στην πλατφόρμα, έχει τη δυνατότητα να δημιουργήσει το προφίλ του, να προσθέσει βιβλία που έχει ήδη διαβάσει και να λάβει προτάσεις βασισμένες στις αξιολογήσεις του. Το σύστημα αναλύει χαρακτηριστικά όπως ο τόνος, ο ρυθμός και η θεματολογία των βιβλίων προκειμένου να παράγει εξατομικευμένες προτάσεις. Ωστόσο, το StoryGraph παρουσιάζει ορισμένες αδυναμίες. Είναι διαθέσιμο αποκλειστικά στην αγγλική γλώσσα, γεγονός που περιορίζει

σημαντικά την προσβασιμότητά του για χρήστες που δεν είναι εξοικειωμένοι με αυτήν. Επιπλέον, η πλατφόρμα εστιάζει κυρίως σε αγγλόφωνους τίτλους, αφήνοντας εκτός μεγάλο μέρος της διεθνούς βιβλιογραφίας.

Το WhichBook είναι μια διαδικτυακή υπηρεσία που ακολουθεί μια διαφορετική προσέγγιση στην επιλογή βιβλίων. Αντί να βασίζεται σε ιστορικό αναγνώσεων, επιτρέπει στον χρήστη να ορίσει συγκεκριμένες παραμέτρους που αντικατοπτρίζουν τη διάθεση και τις προτιμήσεις του, όπως η συναισθηματική ένταση, η πολυπλοκότητα της πλοκής ή ο βαθμός ρεαλισμού του βιβλίου. Με βάση αυτές τις παραμέτρους, η πλατφόρμα προτείνει τίτλους που ταιριάζουν στο προφίλ που όρισε ο χρήστης. Παρά την πρωτότυπη αυτή προσέγγιση, το WhichBook παρουσιάζει ένα βασικό μειονέκτημα: απαιτεί από τον χρήστη να γνωρίζει εκ των προτέρων τι ακριβώς αναζητά, κάτι που δεν ισχύει πάντοτε, ιδίως για χρήστες που δεν έχουν σαφή εικόνα των αναγνωστικών τους προτιμήσεων. Επιπροσθέτως, η πλατφόρμα δεν αξιοποιεί ιστορικό αξιολογήσεων, με αποτέλεσμα να μην είναι δυνατή η εξατομίκευση των προτάσεων με βάση τις προηγούμενες αναγνωστικές εμπειρίες του χρήστη. Το WhatShouldIReadNext αποτελεί μια απλούστερη υπηρεσία, στην οποία ο χρήστης εισάγει έναν τίτλο βιβλίου που του άρεσε και λαμβάνει μια λίστα παρόμοιων προτάσεων βασισμένων στις επιλογές άλλων χρηστών με κοινές προτιμήσεις. Η υπηρεσία αυτή είναι ιδιαίτερα εύχρηστη και προσιτή, καθώς δεν απαιτεί εγγραφή ή δημιουργία προφίλ. Εντούτοις, το γεγονός αυτό αποτελεί ταυτόχρονα και το μεγαλύτερο μειονέκτημά της. Χωρίς προφίλ χρήστη και ιστορικό αξιολογήσεων, οι προτάσεις που παράγονται δεν είναι εξατομικευμένες, αλλά βασίζονται αποκλειστικά στις επιλογές της κοινότητας. Επιπλέον, η πλατφόρμα δεν αξιοποιεί τεχνικές μηχανικής μάθησης, γεγονός που περιορίζει σημαντικά την ακρίβεια και την ποιότητα των αποτελεσμάτων που προσφέρει.

Συμπερασματικά, παρατηρείται ότι τα υπάρχοντα λογισμικά, αν και προσφέρουν χρήσιμες λειτουργίες, δεν καταφέρνουν να συνδυάσουν πλήρως την εξατομίκευση, την αξιοποίηση τεχνικών μηχανικής μάθησης και την απλότητα χρήσης σε μια ενιαία πλατφόρμα. Κρίνεται, επομένως, αναγκαία η δημιουργία μιας πιο ολοκληρωμένης εφαρμογής, η οποία να αντιμετωπίζει τις αδυναμίες αυτές και να προσφέρει στον χρήστη μια πλήρη και αποτελεσματική αναγνωστική εμπειρία. Η ανάγκη αυτή αποτέλεσε το βασικό κίνητρο για την ανάπτυξη της εφαρμογής BookLovers.

1.3 Σκοπός και στόχοι της εργασίας

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός λογισμικού που να ανταποκρίνεται πλήρως στις απαιτήσεις που έχουν οι αναγνώστες βιβλίων προκειμένου να λύσουν ένα σημαντικό πρόβλημα που αντιμετωπίζουν «ποιο βιβλίο να διαβάσω τώρα?» αλλά και να φέρει τον υπόλοιπο κόσμο ποιο κοντά στην ανάγνωση βιβλίων.

Η διαδικτυακή αυτή εφαρμογή αξιοποιεί τεχνικές μηχανικής μάθησης προκειμένου να παρέχει εξατομικευμένες προτάσεις βιβλίων στους χρήστες της, συμβάλλοντας στην αντιμετώπιση των προβλημάτων που αναλύθηκαν. Για την επίτευξη του παραπάνω σκοπού κρίθηκε αναγκαία η υλοποίηση μιας σειράς επιμέρους στόχων. Αρχικά, δημιουργήθηκε ένα ολοκληρωμένο σύστημα διαχείρισης χρηστών, το οποίο παρέχει τη δυνατότητα εγγραφής, σύνδεσης και αποσύνδεσης, ώστε κάθε χρήστης να διαθέτει το δικό του ασφαλές και εξατομικευμένο περιβάλλον. Παράλληλα, υλοποιήθηκε ένα σύστημα αναζήτησης βιβλίων που επιτρέπει στον χρήστη να εντοπίζει τίτλους βάσει διαφόρων κριτηρίων, όπως ο τίτλος ή ο συγγραφέας. Ένας ακόμη κεντρικός στόχος της εφαρμογής είναι η καταγραφή και αξιοποίηση της συμπεριφοράς κάθε χρήστη εντός της πλατφόρμας, ώστε να διαμορφώνεται ένα δυναμικό προφίλ προτιμήσεων που αντικατοπτρίζει τα πραγματικά του ενδιαφέροντα. Βάσει του προφίλ αυτού, το σύστημα επιδιώκει να παράγει προτάσεις βιβλίων που είναι όσο το δυνατόν πιο στοχευμένες και συναφείς με τις ανάγκες του εκάστοτε χρήστη.

Τέλος, ιδιαίτερη έμφαση δόθηκε στη σχεδίαση ενός φιλικού προς τον χρήστη γραφικού περιβάλλοντος, το οποίο παρουσιάζει τις προτάσεις με τρόπο κατανοητό και ελκυστικό, συνοδευόμενες από σαφή αιτιολόγηση, ώστε ο χρήστης να κατανοεί γιατί του προτείνεται κάθε βιβλίο. Συμπερασματικά, η εφαρμογή BookLovers επιδιώκει να αποτελέσει ένα ολοκληρωμένο εργαλείο που προσεγγίζει το πρόβλημα της επιλογής βιβλίου με έναν έξυπνο και εξατομικευμένο τρόπο, θέτοντας τον χρήστη στο επίκεντρο της εμπειρίας.

1.4 Χρήση Εργαλείων Λογισμικού

Για την υλοποίηση της εφαρμογής BookLovers αξιοποιήθηκε συνδυασμός σύγχρονων εργαλείων και τεχνολογιών λογισμικού, η επιλογή των οποίων πραγματοποιήθηκε με γνώμονα την αποτελεσματικότητα, την ευελιξία και την καταλληλόλητά τους για την ανάπτυξη διαδικτυακών εφαρμογών , τα λεγόμενα ως και websites .Ως κύριο περιβάλλον ανάπτυξης χρησιμοποιήθηκε το PyCharm, ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού της JetBrains, ειδικά σχεδιασμένο για προγραμματισμό σε Python. Το PyCharm παρέχει πληθώρα δυνατοτήτων, όπως αυτόματη συμπλήρωση κώδικα, εντοπισμός σφαλμάτων, και ενσωματωμένο εργαλείο αποσφαλμάτωσης, τα οποία συνέβαλαν σημαντικά στην ταχύτερη και αποδοτικότερη ανάπτυξη της εφαρμογής.
Η εφαρμογή αναπτύχθηκε στη γλώσσα προγραμματισμού Python, με τη χρήση του μικροπλαισίου Flask. Το Flask αποτελεί ένα ελαφρύ και ευέλικτο πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών, το οποίο επιτρέπει την ανάπτυξη εφαρμογών με απλή και κατανοητή δομή, παρέχοντας παράλληλα όλα τα απαραίτητα εργαλεία για τη διαχείριση των αιτημάτων, των διαδρομών και των προτύπων της εφαρμογής.
Για τη διαχείριση των δεδομένων της εφαρμογής επιλέχθηκε το SQLite, ένα ελαφρύ σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που δεν απαιτεί ξεχωριστό διακομιστή για τη λειτουργία του, και μπορεί να εγκατασταθεί μέσα στο PyCharm Project από την γραμμή εντολών που διαθέτει. Η επικοινωνία με τη βάση δεδομένων πραγματοποιείται μέσω της βιβλιοθήκης Flask-SQLAlchemy, η οποία παρέχει ένα αντικειμενοστρεφές επίπεδο αλληλεπίδρασης με τη βάση, απλοποιώντας σημαντικά τη διαχείριση των οντοτήτων και των σχέσεων τους. Για την υλοποίηση του συστήματος προτάσεων αξιοποιήθηκε η βιβλιοθήκη scikit-learn, η οποία παρέχει έτοιμες υλοποιήσεις αλγορίθμων μηχανικής μάθησης. Συγκεκριμένα, χρησιμοποιήθηκαν οι τεχνικές TF-IDF διανυσματοποίησης και cosine similarity για τον υπολογισμό της ομοιότητας μεταξύ του προφίλ του χρήστη και των υποψήφιων βιβλίων, αποτελώντας τον πυρήνα του συστήματος content-based filtering της εφαρμογής. Για την ανάκτηση βιβλιογραφικών δεδομένων σε πραγματικό χρόνο αξιοποιήθηκε το Open Library API, μια ανοιχτή διαδικτυακή υπηρεσία που παρέχει πρόσβαση σε εκατομμύρια τίτλους βιβλίων με πληροφορίες όπως τίτλος, συγγραφέας, εξώφυλλο και αξιολογήσεις. Η επικοινωνία με το API πραγματοποιείται μέσω της βιβλιοθήκης Requests της Python.
Τέλος, για τη σχεδίαση του γραφικού περιβάλλοντος της εφαρμογής χρησιμοποιήθηκαν οι τεχνολογίες HTML και CSS, μέσω των οποίων υλοποιήθηκαν οι σελίδες της εφαρμογής με φιλικό και λειτουργικό σχεδιασμό. Η δημιουργία δυναμικών σελίδων πραγματοποιήθηκε με τη χρήση της μηχανής προτύπων Jinja2, η οποία είναι ενσωματωμένη στο Flask σαν εντολή και επιτρέπει την ευέλικτη παραγωγή HTML περιεχομένου βάσει των δεδομένων της εφαρμογής.

Κεφάλαιο 2ο

2 Σχεδίαση της εφαρμογής

Στο κεφάλαιο αυτό του παρόντος επιστημονικού εγγράφου πραγματεύεται η ανάλυση και σχεδίαση της διαδικτυακής εφαρμογής μας BookLovers, ενέργειες δηλαδη που κατέχουν σημαντικό ρόλο και πρέπει να πραγματοποιηθούν προτού προχωρήσουμε σε μια ολοκληρωμένη υλοποίηση εφαρμογής στον κλάδο της Πληροφορικής. Η ανάλυση των απαιτήσεων δηλαδή που θα πραγματοποιηθεί θα μας βοηθήσει στο να μειωθεί σημαντικά η πολυπλοκότητα του συστήματος που θέλουμε να κατασκευάσουμε και να είναι δυνατή η σχεδίαση του. Αφετηρία, λοιπόν, της σχεδίασης μιας τέτοιου είδους εφαρμογής αποτελεί η δημιουργία ορισμένων διαγραμμάτων που κάνουν πιο κατανοητή τη συνολική λειτουργία της και τις απαιτήσεις της. Τέτοια διαγράμματα είναι τα γνωστά διαγράμματα UML , Unified Modeling Language , και αποτελούν επί της ουσίας μια γλώσσα μοντελοποίησης ενός πληροφοριακού συστήματος. Συνεπώς, όπως γίνεται αντιληπτό, με τη χρήση των διαγραμμάτων αυτών είναι ευκολότερο τόσο στον προγραμματιστή να δημιουργήσει μια πολύπλοκη εφαρμογή όσο και στον απλό χρήστη να κατανοήσει τη λειτουργία της, χωρίς να γνωρίζει απαραίτητα το τεχνικό κομμάτι που υπάρχει πίσω από αυτή. Φυσικά, εξίσου σημαντικό ρόλο σε μια εφαρμογή Πληροφορικής κατέχει και η σχεδίαση της βάσης δεδομένων αυτής.

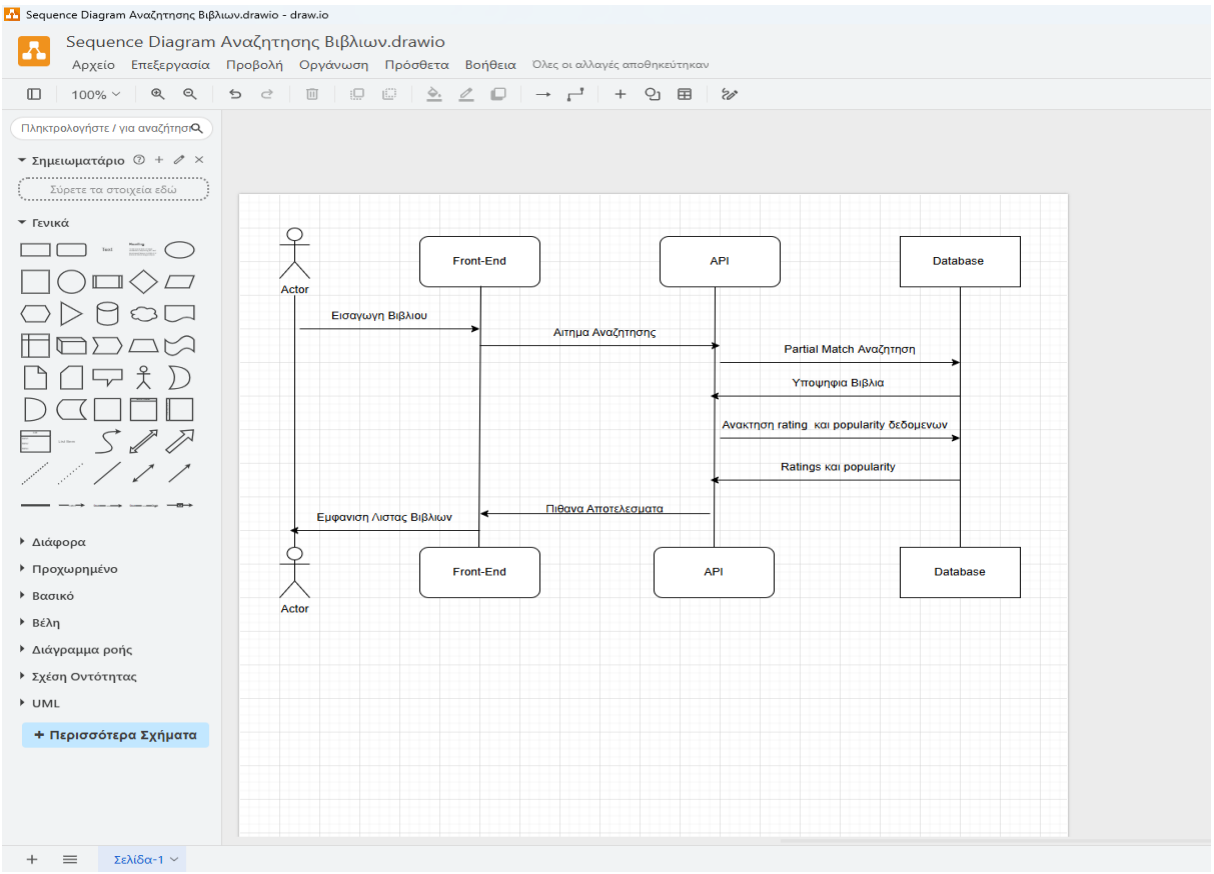
2.1 Χρήση ακολουθιακών διαγραμμάτων UML

Τα ακολουθιακά διαγράμματα (Sequence Diagrams) αποτελούν έναν από τους βασικότερους τύπους διαγραμμάτων της Unified Modeling Language (UML) και χρησιμοποιούνται για την απεικόνιση της δυναμικής συμπεριφοράς ενός συστήματος. Σκοπός τους είναι να παρουσιάσουν τη χρονική ακολουθία των αλληλεπιδράσεων μεταξύ των συμμετεχόντων (αντικειμένων, κλάσεων ή υποσυστημάτων) κατά την εκτέλεση μιας συγκεκριμένης λειτουργίας ή διαδικασίας.Τα ακολουθιακά διαγράμματα χρησιμοποιούνται ευρέως κατά την ανάλυση και τον σχεδιασμό πληροφοριακών συστημάτων, καθώς επιτρέπουν την αναλυτική απεικόνιση της ροής εκτέλεσης μιας λειτουργίας και των αλληλεπιδράσεων μεταξύ των επιμέρους στοιχείων του συστήματος. Επιπλέον, διευκολύνουν την κατανόηση των λειτουργικών απαιτήσεων, την επικοινωνία μεταξύ αναλυτών και προγραμματιστών, καθώς και τον εντοπισμό πιθανών σφαλμάτων ή ασυνεπειών στη σχεδίαση πριν από την υλοποίηση. Βασικά στοιχεία ενός ακολουθιακού διαγράμματος αποτελούν οι συμμετέχοντες (actors ή objects), οι γραμμές ζωής (lifelines), οι ενεργοποιήσεις (activation bars) που υποδηλώνουν την εκτέλεση μιας ενέργειας και τα μηνύματα (messages), τα οποία αποτυπώνουν τη σειρά και τον τρόπο επικοινωνίας μεταξύ των συμμετεχόντων. Χάρη στη δυνατότητα απεικόνισης της χρονικής ακολουθίας των γεγονότων, τα ακολουθιακά διαγράμματα αποτελούν ένα ιδιαίτερα χρήσιμο εργαλείο για τον σχεδιασμό εφαρμογών λογισμικού και πληροφοριακών συστημάτων. Πιο συγκεκριμένα τα ακολουθιακά διαγράμματα που παρουσιάζουμε είναι τρία:

- Sequence Diagram Αναζήτησης βιβλίων
- Sequence Diagram Καταγραφή Ενεργειών Χρήστη
- Sequence Diagram Εξατομικευμένων Προτάσεων Βιβλίων

2.1.1 Ακολουθιακό διάγραμμα αναζήτησης βιβλίων

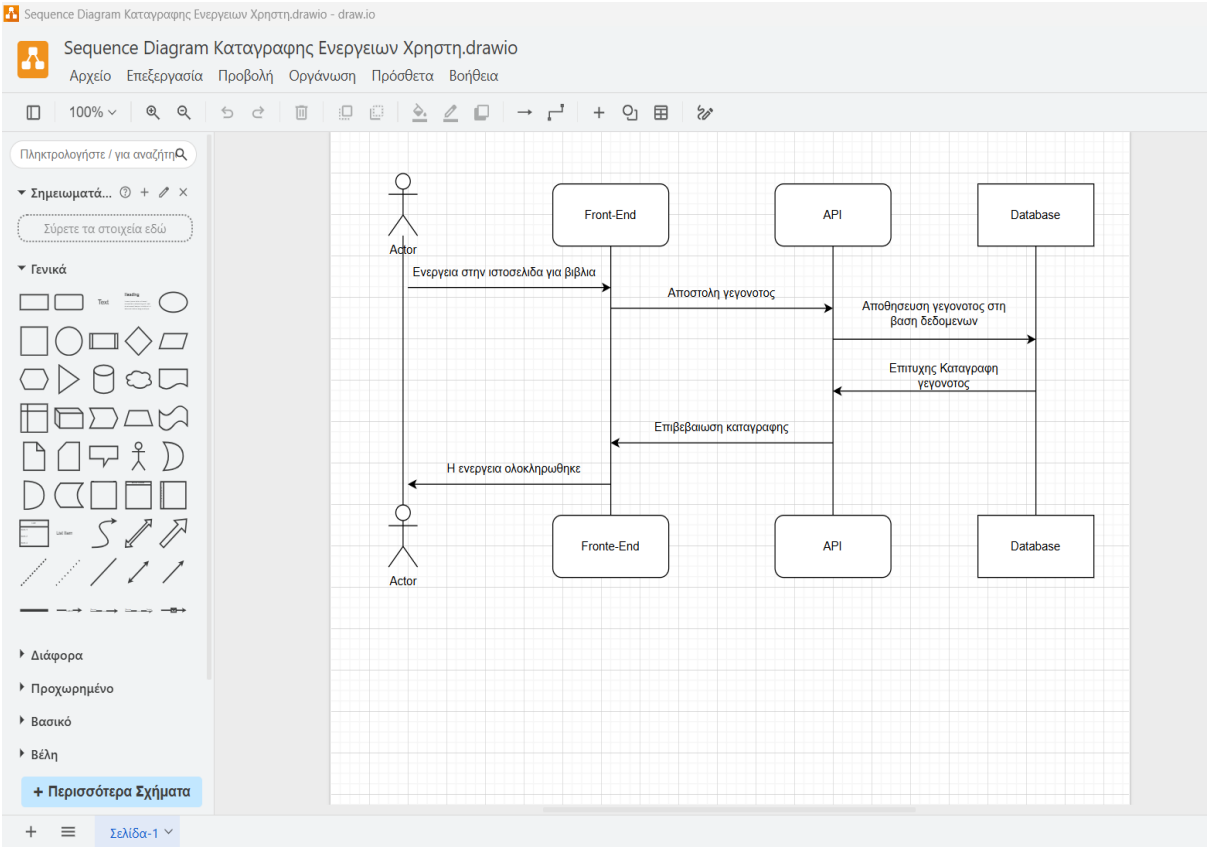
Το διάγραμμα ακολουθίας αναζήτησης βιβλίων απεικονίζει τη διαδικασία που ακολουθείται όταν ένας χρήστης επιθυμεί να αναζητήσει ένα βιβλίο μέσω της εφαρμογής. Στόχος του διαγράμματος είναι να παρουσιάσει τη χρονική σειρά των αλληλεπιδράσεων μεταξύ του χρήστη (Actor), του περιβάλλοντος διεπαφής (Front-End), του διακομιστή εφαρμογής (API) και της βάσης δεδομένων (Database), μέχρι την εμφάνιση των αποτελεσμάτων αναζήτησης.
Η διαδικασία ξεκινά με την εισαγωγή του ονόματος ή μέρους του τίτλου του βιβλίου από τον χρήστη στη γραμμή αναζήτησης της εφαρμογής. Το Front-End λαμβάνει την είσοδο του χρήστη και αποστέλλει αίτημα αναζήτησης (Search Request) προς το API, το οποίο είναι υπεύθυνο για τη διαχείριση και την επεξεργασία του αιτήματος. Στη συνέχεια, το API επικοινωνεί με τη βάση δεδομένων και εκτελεί αναζήτηση τύπου Partial Match, προκειμένου να εντοπίσει βιβλία που ταιριάζουν πλήρως ή μερικώς με το κείμενο που εισήγαγε ο χρήστης. Αφού εντοπιστούν τα υποψήφια βιβλία, το API ζητά επιπλέον πληροφορίες από τη βάση δεδομένων σχετικά με τη βαθμολογία (rating) και τη δημοτικότητα (popularity) κάθε βιβλίου. Με την ολοκλήρωση της επεξεργασίας, η βάση δεδομένων επιστρέφει στο API τα απαραίτητα στοιχεία, το οποίο δημιουργεί τη λίστα των πιθανών αποτελεσμάτων και την αποστέλλει στο Front-End. Τέλος, το Front-End παρουσιάζει στον χρήστη τη λίστα των βιβλίων που αντιστοιχούν στην αναζήτησή του, επιτρέποντάς του να επιλέξει το επιθυμητό βιβλίο ή να πραγματοποιήσει νέα αναζήτηση.



Εικόνα 1 Διάγραμμα αναζήτησης βιβλίων

2.1.2 Ακολουθιακό διάγραμμα καταγραφής ενεργειών χρήστη

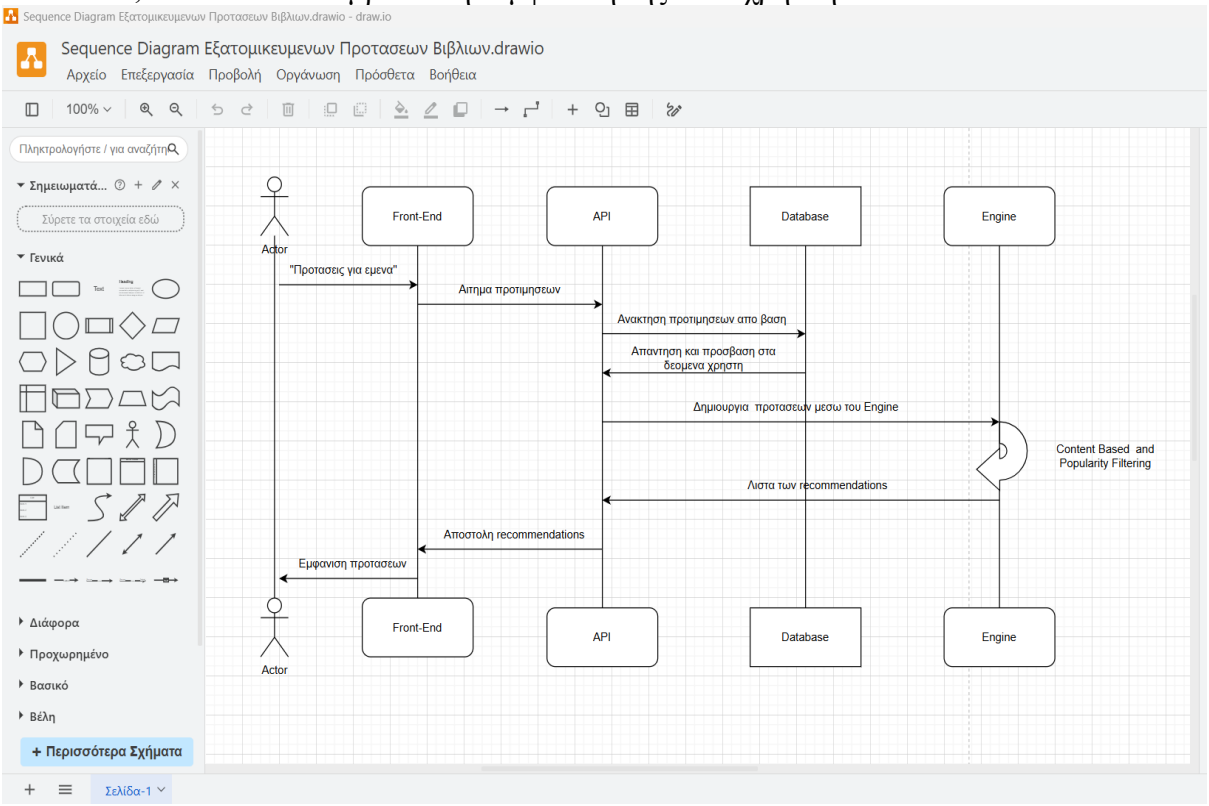
Το ακολουθιακό διάγραμμα καταγραφής ενεργειών χρήστη αποτυπώνει τη ροή αλληλεπίδρασης μεταξύ των βασικών συνιστωσών του συστήματος κατά τη διάρκεια μιας χαρακτηριστικής ενέργειας εντός της ιστοσελίδας. Ειδικότερα, το διάγραμμα περιγράφει τον τρόπο με τον οποίο κάθε ενέργεια του χρήστη ανιχνεύεται, μεταδίδεται και αποθηκεύεται, διαμορφώνοντας έτσι ένα αξιόπιστο σύστημα συλλογής δεδομένων συμπεριφοράς.
Στο διάγραμμα συμμετέχουν τέσσερις οντότητες: ο Χρήστης, το Front-End, το API και η Βάση Δεδομένων. Η ακολουθία εκκινεί όταν ο χρήστης πραγματοποιεί μια ενέργεια που σχετίζεται με ένα βιβλίο εντός της ιστοσελίδας. Οι ενέργειες αυτές ενδέχεται να είναι η προβολή ή το κλικ σε ένα βιβλίο αλλά ενέργειες όπως η αξιολόγησή του ή η προσθήκη του στη λίστα επιθυμιών. Στο πρώτο βήμα, το Front-End εντοπίζει την ενέργεια του χρήστη και την επεξεργάζεται τοπικά, δομώντας την σε κατάλληλη αναπαράσταση γεγονότος. Στη συνέχεια, αποστέλλει το γεγονός αυτό προς το API, το οποίο αναλαμβάνει την επικοινωνία με τη βάση δεδομένων. Το API, με τη σειρά του, προωθεί το γεγονός στη βάση δεδομένων για μόνιμη αποθήκευση. Η βάση δεδομένων επιστρέφει επιβεβαίωση επιτυχούς καταγραφής, την οποία το API διαβιβάζει στο Front-End. Τέλος, το Front-End ενημερώνει τον χρήστη ότι η ενέργεια ολοκληρώθηκε επιτυχώς. Τα δεδομένα που συλλέγονται μέσω αυτής της ροής αποτελούν τη βασική πηγή πληροφορίας για τη δημιουργία του προφίλ κάθε χρήστη. Πιο συγκεκριμένα, ενέργειες όπως view, clicked, added to wishlist, read και rated αξιοποιούνται από τον μηχανισμό προτάσεων της εφαρμογής, τροφοδοτώντας αλγορίθμους Collaborative ή Content-Based Filtering με πραγματικά δεδομένα συμπεριφοράς.



Εικόνα 2 Διάγραμμα καταγραφής ενεργειών χρήστη

2.1.3 Ακολουθιακό διάγραμμα εξατομικευμένων προτάσεων βιβλίων

Το ακολουθιακό διάγραμμα εξατομικευμένων προτάσεων βιβλίων αποτυπώνει τη λεπτομερή ροή επικοινωνίας μεταξύ των συνιστωσών του συστήματος κατά την εκτέλεση της βασικής λειτουργίας του recommendation engine. Στο διάγραμμα συμμετέχουν πέντε οντότητες: ο Χρήστης, το Front-End, το API, η Βάση Δεδομένων και ο Μηχανισμός Προτάσεων (Engine). Η ακολουθία εκκινεί όταν ο χρήστης επιλέξει τη λειτουργία «Recommendations» εντός της ιστοσελίδας. Το Front-End λαμβάνει το αίτημα και το προωθεί στο API, ζητώντας την ανάκτηση των αποθηκευμένων προτιμήσεων του συγκεκριμένου χρήστη. Το API, με τη σειρά του, απευθύνεται στη Βάση Δεδομένων, απ' όπου ανακτά το ιστορικό ενεργειών και αλληλεπιδράσεων που έχει καταγραφεί μέσω της διαδικασίας που περιγράφηκε στο προηγούμενο ακολουθιακό διάγραμμα. Η Βάση Δεδομένων επιστρέφει τα δεδομένα του χρήστη στο API, παρέχοντας πρόσβαση σε όλο το αναγκαίο υλικό για την παραγωγή προτάσεων. Στο επόμενο βήμα, το API διαβιβάζει τα δεδομένα αυτά στον Μηχανισμό Προτάσεων ο οποίος αναλαμβάνει την παραγωγή εξατομικευμένων αποτελεσμάτων. Εντός αυτού εκτελείται ένα εσωτερικό βρόχο επεξεργασίας, κατά το οποίο εφαρμόζονται διαφορετικές τεχνικές φιλτραρίσματος. Συγκεκριμένα, χρησιμοποιούνται Content-Based Filtering, καθώς και Popularity Filtering. Μόλις ολοκληρωθεί η επεξεργασία, ο Engine επιστρέφει στο API μια διατεταγμένη λίστα προτάσεων. Το API διαβιβάζει τη λίστα αυτή στο Front-End, το οποίο αναλαμβάνει την εμφάνισή της στον χρήστη.



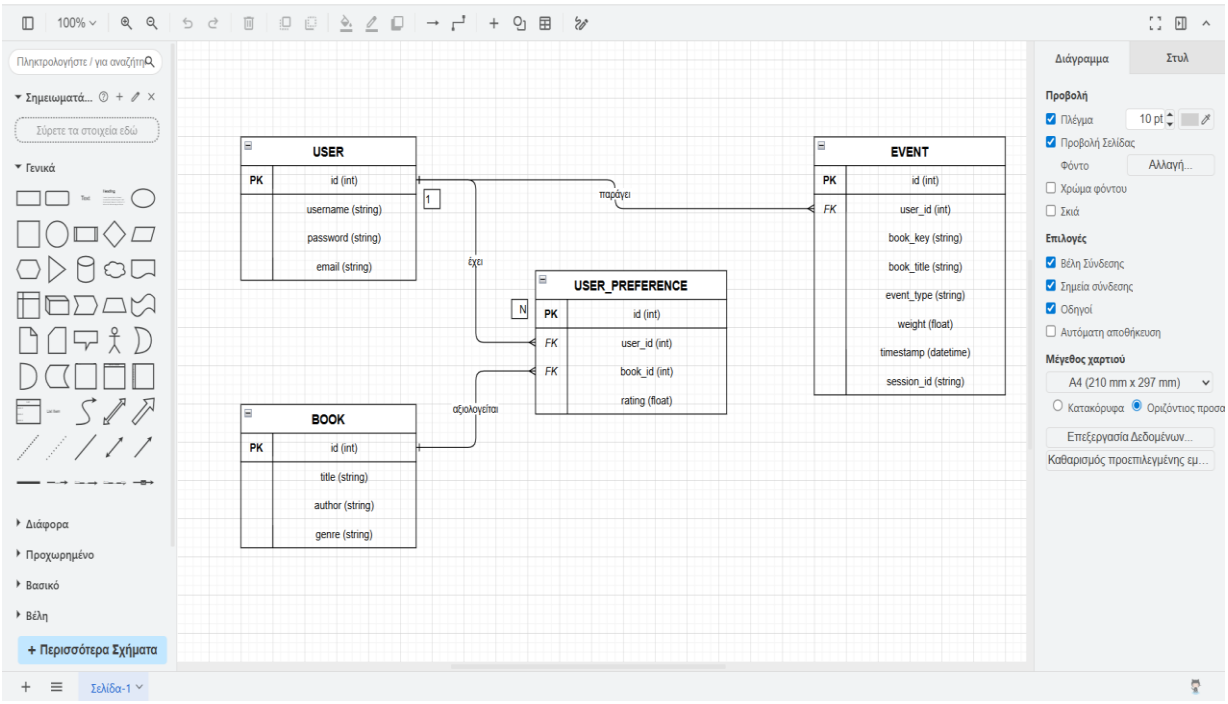
Εικόνα 3 Διάγραμμα εξατομικευμένων προτάσεων βιβλίων

2.2 Δημιουργία βάσης δεδομένων

Η βάση δεδομένων όπως γνωρίζουμε αποτελεί ένα από τα σημαντικότερα χαρακτηριστικά μιας οποιασδήποτε ψηφιακής εφαρμογής μιας και σε αυτήν αποθηκεύεται και συλλέγεται μεγάλος όγκος δεδομένων. Η υλοποίησή της πραγματοποιείται με τον αλληλο-συσχετισμό αρχείων δεδομένων έτσι ώστε να επιτυγχάνεται η προσπέλασή τους όσο το δυνατόν ταχύτερα. Η βάση δεδομένων του παρόντος λογισμικού ονομάζεται BookLovers.db και η διαχείριση της γίνεται μέσα από την εφαρμογή PyCharm, κατεβάζοντας από το Command Prompt (γραμμή εντολών) τις κατάλληλες βιβλιοθήκες. Ο τύπος βάσης που χρησιμοποιείται είναι γνωστός και ως SQLite.

2.2.1 Μοντέλο Οντοτήτων Συσχετίσεων (Entity-Relationship Model)

Ακολούθως παρουσιάζεται το Entity Relationship διάγραμμα της βάσης δεδομένων, ώστε να γίνει αντιληπτός ο τρόπος με τον οποίο οι οντότητες αυτής συνδέονται μεταξύ τους αλλά και το πως λειτουργεί το σύνολο του μοντέλου που αποτελεί τη βάση δεδομένων της διαδικτυακής εφαρμογής.



Εικόνα 4 Μοντέλο οντοτήτων συσχετίσεων βάσης δεδομένων

Στο παραπάνω διάγραμμα , που είναι σχεδιασμένο μέσω της εφαρμογής drawio , απεικονίζονται όλοι οι πίνακες της βάσης δεδομένων μας.

Πίνακας USER

Ο πίνακας USER αναπαριστά τους εγγεγραμμένους η μη χρήστες της εφαρμογής, συμπεριλαμβανομένων και των προκαθορισμένων χρηστών που έχουν δημιουργηθεί εκ των προτέρων στο σύστημα, και αποτελείται από τις εξής στήλες:

- *id* (τύπου int): Αφορά τον μοναδικό αναγνωριστικό αριθμό ,η αλλιώς Primary Key, κάθε χρήστη και αποτελεί το πρωτεύον κλειδί του πίνακα.

- *username* (τύπου string): Αφορά το μοναδικό όνομα χρήστη που επιλέγεται κατά τη διαδικασία εγγραφής στην εφαρμογή.
- *password* (τύπου string): Αφορά τον κωδικό πρόσβασης του χρήστη, ο οποίος αποθηκεύεται σε hash μορφή όπως μια Live Βάση Δεδομενων
- *email* (τύπου string): Αφορά τη διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη, η οποία χρησιμοποιείται κατά τη διαδικασία εγγραφής και σύνδεσης.

Ο παραπάνω πίνακας συνδέεται με τους υπόλοιπους πίνακες της βάσης δεδομένων μέσω του πεδίου *id*. Πιο συγκεκριμένα, συνδέεται με τον πίνακα USER_PREFERENCE μέσω του πεδίου *user_id* και με τον πίνακα EVENT μέσω του ομώνυμου πεδίου *user_id*.

Πίνακας BOOK

Ο πίνακας BOOK αναπαριστά τα βιβλία με τα οποία αλληλεπιδρούν οι χρήστες εντός της εφαρμογής και αποτελείται από τις εξής στήλες:

- *id* (τύπου int): Αφορά τον μοναδικό αναγνωριστικό αριθμό ,η αλλιώς Primary Key, κάθε βιβλίου και αποτελεί το πρωτεύον κλειδί του πίνακα.
- *title* (τύπου string): Αφορά τον τίτλο του βιβλίου όπως αυτός επιστρέφεται από το Open Library API.
- *author* (τύπου string): Αφορά το όνομα του συγγραφέα του βιβλίου.
- *genre* (τύπου string): Αφορά το λογοτεχνικό είδος στο οποίο ανήκει το βιβλίο.

Ο παραπάνω πίνακας συνδέεται με τον πίνακα USER_PREFERENCE μέσω του πεδίου *book_id*, το οποίο αποτελεί ξένο κλειδί που παραπέμπει στο πρωτεύον κλειδί *id* του παρόντος πίνακα. Δεν συνδέεται με τον πίνακα EVENT, καθώς ο τελευταίος αποθηκεύει τα στοιχεία του βιβλίου απευθείας μέσω του *book_key* που επιστρέφεται από το Open Library API.

Πίνακας USER_PREFERENCE

Ο πίνακας USER_PREFERENCE λειτουργεί ως πίνακας σύνδεσης μεταξύ των πινάκων USER και BOOK, καταγράφοντας τις αξιολογήσεις που αποδίδουν οι χρήστες στα βιβλία που έχουν διαβάσει, και αποτελείται από τις εξής στήλες:

- *id* (τύπου int): Αφορά τον μοναδικό αναγνωριστικό αριθμό κάθε εγγραφής και αποτελεί το πρωτεύον κλειδί του πίνακα.
- *user_id* (τύπου int): Αφορά τον αναγνωριστικό αριθμό του χρήστη που πραγματοποίησε την αξιολόγηση και αποτελεί ξένο κλειδί που παραπέμπει στον πίνακα USER.
- *book_id* (τύπου int): Αφορά τον αναγνωριστικό αριθμό του βιβλίου που αξιολογήθηκε και αποτελεί ξένο κλειδί που παραπέμπει στον πίνακα BOOK.
- *rating* (τύπου float): Αφορά τη βαθμολογία που απέδωσε ο χρήστης στο βιβλίο και χρησιμοποιείται από τον αλγόριθμο προτάσεων για τον υπολογισμό της δημοτικότητας κάθε βιβλίου.

Ο παραπάνω πίνακας συνδέεται με τον πίνακα USER μέσω του πεδίου *user_id* και με τον πίνακα BOOK μέσω του πεδίου *book_id*.

Πίνακας EVENT

Ο πίνακας EVENT αποτελεί τον κεντρικό πυρήνα του συστήματος προτάσεων, καθώς καταγράφει κάθε ενέργεια που πραγματοποιεί ο χρήστης εντός της εφαρμογής, και αποτελείται από τις εξής στήλες:

- *id* (τύπου int): Αφορά τον μοναδικό αναγνωριστικό αριθμό κάθε ενέργειας και αποτελεί το πρωτεύον κλειδί του πίνακα.

- *user_id* (τύπου int): Αφορά τον αναγνωριστικό αριθμό του χρήστη που πραγματοποίησε την ενέργεια και αποτελεί ξένο κλειδί που παραπέμπει στον πίνακα USER.
- *book_key* (τύπου string): Αφορά το μοναδικό αναγνωριστικό του βιβλίου όπως αυτό επιστρέφεται από το Open Library API.
- *book_title* (τύπου string): Αφορά τον τίτλο του βιβλίου και αποθηκεύεται απευθείας για άμεση αναφορά χωρίς την ανάγκη επιπλέον ερωτημάτων στη βάση δεδομένων.
- *event_type* (τύπου string): Αφορά τον τύπο της ενέργειας που πραγματοποίησε ο χρήστης, όπως αναζήτηση, κλικ ή προσθήκη στη λίστα επιθυμιών.
- *weight* (τύπου float): Αφορά το αριθμητικό βάρος που αποδίδεται σε κάθε τύπο ενέργειας, καθώς διαφορετικές ενέργειες σηματοδοτούν διαφορετικό επίπεδο ενδιαφέροντος του χρήστη για ένα βιβλίο.
- *timestamp* (τύπου datetime): Αφορά την ακριβή χρονική στιγμή κατά την οποία πραγματοποιήθηκε η ενέργεια.
- *session_id* (τύπου string): Αφορά το αναγνωριστικό της συνεδρίας εντός της οποίας εκτελέστηκε η ενέργεια.

Ο παραπάνω πίνακας συνδέεται με τον πίνακα USER μέσω του πεδίου *user_id*. Δεν συνδέεται άμεσα με τον πίνακα BOOK, καθώς τα στοιχεία του βιβλίου αποθηκεύονται απευθείας μέσω του *book_key* που επιστρέφεται από το Open Library API.

Κεφάλαιο 3ο

3 Αναλυτική περιγραφή της υλοποίησης της εφαρμογής

Σε αυτό το κεφάλαιο θα πραγματοποιηθεί αναλυτική περιγραφή της υλοποίησης της διαδικτυακής εφαρμογής που δημιουργήθηκε. Σκοπός είναι να γίνει πλήρως κατανοητός ο τρόπος με τον οποίο υλοποιήθηκε η ιστοσελίδα BookLovers, λαμβάνοντας υπόψιν τόσο τη δομή της βάσης δεδομένων όσο και τις τεχνολογίες που παρουσιάστηκαν στα προηγούμενα κεφάλαια. Τα στοιχεία αυτά αποτελούν τη βάση επάνω στην οποία στηρίχτηκε η σχεδίαση της εφαρμογής και τελικά έγινε η μετάβαση από τη θεωρία στην πράξη.
Στο παρόν κεφάλαιο θα αναλυθούν διεξοδικά όλες οι λειτουργίες που προσφέρει η εφαρμογή, ξεκινώντας από την εφαρμογή από την πλευρά του μη εγγεγραμμένου χρήστη και φτάνοντας στον πυρήνα της εφαρμογής, δηλαδή στο σύστημα παραγωγής εξατομικευμένων προτάσεων βιβλίων. Παράλληλα, θα περιγράψουμε τον τρόπο με τον οποίο η εφαρμογή καταγράφει τη συμπεριφορά του χρήστη, αξιοποιεί τα δεδομένα αυτά μέσω του υβριδικού αλγορίθμου μηχανικής μάθησης και παράγει τελικά προτάσεις που ανταποκρίνονται στις ιδιαίτερες αναγνωστικές προτιμήσεις του κάθε χρήστη.

3.1 Η εφαρμογή από την πλευρά του μη εγγεγραμμένου χρήστη

Η διαδικτυακή εφαρμογή BookLovers είναι σχεδιασμένη κατά τρόπο που να επιτρέπει στον επισκέπτη της πλατφόρμας να αλληλεπιδρά με ορισμένες βασικές λειτουργίες της, χωρίς να απαιτείται η δημιουργία λογαριασμού. Συγκεκριμένα, ένας μη εγγεγραμμένος χρήστης έχει τη δυνατότητα να πλοηγηθεί στην αρχική σελίδα της εφαρμογής και να πραγματοποιήσει αναζήτηση βιβλίων μέσω του Open Library API, προβάλλοντας τα αποτελέσματα που επιστρέφει το σύστημα. Ωστόσο, οι δυνατότητες αυτές είναι περιορισμένες σε σύγκριση με αυτές που προσφέρονται στον εγγεγραμμένο χρήστη, καθώς λειτουργίες όπως η αποθήκευση βιβλίων στη λίστα επιθυμιών και η λήψη εξατομικευμένων προτάσεων απαιτούν την ύπαρξη ενεργού λογαριασμού στην πλατφόρμα. Η αρχική σελίδα της εφαρμογής αντιστοιχεί στη διαδρομή '/' και αποτελεί το κεντρικό σημείο εισόδου στην πλατφόρμα. Είναι προσβάσιμη από οποιονδήποτε επισκέπτη, ανεξαρτήτως εάν διαθέτει λογαριασμό ή όχι. Σκοπός της είναι να παρουσιάσει στον χρήστη τις βασικές δυνατότητες της εφαρμογής και να τον καθοδηγήσει είτε προς τη σελίδα αναζήτησης βιβλίων είτε προς τη σελίδα σύνδεσης και εγγραφής. Το αντίστοιχο κομμάτι κώδικα που υλοποιεί την αρχική σελίδα και την αρχικοποίηση της εφαρμογής στο αρχείο app.py είναι το ακόλουθο:

Κώδικα από το αρχείο app.py: Αρχικοποίηση εφαρμογής και αρχική σελίδα

```
from flask import Flask, render_template, url_for, flash, request, redirect, jsonify, session
from modelsDB import db, User, Book, UserPreference, Event
from flask_login import LoginManager, login_user, logout_user, login_required, curre
nt_user
from werkzeug.security import generate_password_hash, check_password_hash
from validate_password_policies import validate_password
from search_api import search_books_openlibrary
from recommendations import get_recommendations
import uuid
from datetime import datetime, timedelta
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///BookLovers.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.secret_key = 'BookLovers-secret-key-2025-diploma'
app.config['REMEMBER_COOKIE_DURATION'] = timedelta(days=7)
app.config['SESSION_COOKIE_SAMESITE'] = 'Lax'
app.config['SESSION_COOKIE_SECURE'] = False
db.init_app(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'
@login_manager.user_loader
def load_user(user_id):
 return db.session.get(User, int(user_id))
@app.route('/')
def home():
 return render_template('BookStoreMainPage.html')
```

Όπως φαίνεται στον παραπάνω κώδικα, αρχικά εισάγονται όλες οι απαραίτητες βιβλιοθήκες και modules της εφαρμογής. Στη συνέχεια πραγματοποιείται η αρχικοποίηση της εφαρμογής Flask και ο ορισμός των βασικών ρυθμίσεών της. Η ρύθμιση SQLALCHEMY_DATABASE_URI ορίζει τη διαδρομή της βάσης δεδομένων SQLite, ενώ το secret_key χρησιμοποιείται για την κρυπτογράφηση των cookies της συνεδρίας. Η ρύθμιση REMEMBER_COOKIE_DURATION ορίζει ότι τα cookies παραμένουν ενεργά για 7 ημέρες, επιτρέποντας στον χρήστη να παραμένει συνδεδεμένος ακόμα και μετά το κλείσιμο του προγράμματος περιήγησης. Η ρύθμιση SESSION_COOKIE_SAMESITE με τιμή 'Lax' παρέχει προστασία από επιθέσεις τύπου Cross-Site Request Forgery (CSRF).
Στη συνέχεια αρχικοποιείται το αντικείμενο LoginManager της βιβλιοθήκης Flask-Login, το οποίο αναλαμβάνει τη διαχείριση της ταυτοποίησης των χρηστών. Η ρύθμιση login_manager.login_view = 'login' ορίζει ότι εάν ένας μη εγγεγραμμένος χρήστης επιχειρήσει να αποκτήσει πρόσβαση σε προστατευμένη σελίδα, θα ανακατευθυνθεί αυτόματα στη σελίδα σύνδεσης. Η συνάρτηση load_user() διακοσμείται με τον decorator login_manager.user_loader και αναλαμβάνει την ανάκτηση του αντικειμένου χρήστη από τη βάση δεδομένων βάσει του αναγνωριστικού του, κάθε φορά που απαιτείται η ταυτοποίησή του. Τέλος, η διαδρομή '/' αντιστοιχεί στη συνάρτηση home(), η οποία αποδίδει το αρχείο προτύπου BookStoreMainPage.html μέσω της μηχανής προτύπων Jinja2. Μια από τις βασικότερες λειτουργίες που προσφέρει η εφαρμογή BookLovers στον μη εγγεγραμμένο χρήστη είναι η

δυνατότητα αναζήτησης βιβλίων. Ο χρήστης μπορεί να εισάγει έναν όρο αναζήτησης στο αντίστοιχο πεδίο και να λάβει αποτελέσματα από τη διαδικτυακή βιβλιοθήκη Open Library. Η αναζήτηση υποστηρίζει τρία διαφορετικά κριτήρια φιλτραρίσματος: αναζήτηση βάσει τίτλου (title), βάσει συγγραφέα (author) και βάσει λογοτεχνικού είδους (genre). Εάν δεν επιλεγεί κάποιο συγκεκριμένο κριτήριο, η αναζήτηση πραγματοποιείται βάσει γενικού όρου (all). Ο πλήρης κώδικας της συνάρτησης αναζήτησης στο αρχείο app.py είναι ο ακόλουθος:

Κώδικας από το αρχείο app.py : Αναζήτηση βιβλίων

```
@app.route('/search')
def search():
 query = request.args.get('q', '').strip()
 filter_by = request.args.get('filter', 'all')
 books = []
 error = None
 if query:
 log_event(event_type='searched', book_key=f'search:{query}',
 book_title=query, weight=2.0)
 books = search_books_openlibrary(query, filter_by=filter_by, limit=12)
 if not books:
 error = f'Den vrethikan apotelesmata gia "{query}".'
 return render_template('search.html', books=books,
 query=query, filter_by=filter_by, error=error)
```

Η συνάρτηση search() λαμβάνει δύο παραμέτρους από το αίτημα GET: τον όρο αναζήτησης q και το κριτήριο φιλτραρίσματος filter. Εφόσον ο χρήστης εισάγει κάποιο όρο αναζήτησης, το σύστημα καταγράφει αυτόματα ένα γεγονός αναζήτησης (event_type='searched') μέσω της συνάρτησης log_event(), αποδίδοντάς του βάρος 2.0. Η καταγραφή αυτή πραγματοποιείται ακόμα και για μη εγγεγραμμένους χρήστες, με τη διαφορά ότι το πεδίο user_id του γεγονότος λαμβάνει την τιμή None. Στη συνέχεια καλείται η συνάρτηση search_books_openlibrary() από το αρχείο search_api.py, η οποία αναλαμβάνει την επικοινωνία με το Open Library API. Σε περίπτωση που δεν βρεθούν αποτελέσματα εμφανίζεται στον χρήστη κατάλληλο μήνυμα ενημέρωσης. Η επικοινωνία με το Open Library API υλοποιείται στο αρχείο search_api.py μέσω της συνάρτησης search_books_openlibrary(). Ο πλήρης κώδικας της συνάρτησης αυτής παρατίθεται ακολούθως:

Κώδικας από το αρχείο search_api.py : Επικοινωνία με Open Library API:

```
 import requests
from popularity import sort_by_popularity
def search_books_openlibrary(query, filter_by='all', limit=12):
 if not query or not query.strip():
 return []
 base_url = 'https://openlibrary.org/search.json'
 params = {
 'limit': limit,
 'fields': 'key,title,author_name,first_publish_year,
 cover_i,subject,ratings_average,ratings_count'
 }
```

```
if filter_by == 'title':
params['title'] = query
elif filter_by == 'author':
params['author'] = query
elif filter_by == 'genre':
params['subject'] = query
else:
params['q'] = query
try:
response = requests.get(base_url, params=params, timeout=8)
response.raise_for_status()
data = response.json()
books = []
for doc in data.get('docs', []):
cover_id = doc.get('cover_i')
cover_url = (f'https://covers.openlibrary.org/b/id/{cover_id}-M.jpg'
if cover_id else None)
rating = doc.get('ratings_average', 0)
rating = round(rating, 1) if rating else None
subjects = doc.get('subject', [])
books.append({
'key': doc.get('key', ''),
'title': doc.get('title', 'Unknown Title'),
'author': ', '.join(doc.get('author_name',
['Unknown Author'])[:2]),
'year': doc.get('first_publish_year', ''),
'cover_url': cover_url,
'rating': rating,
'ratings_count': doc.get('ratings_count', 0),
'genres': subjects[:3] if subjects else [],
'ol_url': f'https://openlibrary.org{doc.get("key", "")}',
})
books = sort_by_popularity(books)
return books
except requests.exceptions.Timeout:
return []
except requests.exceptions.RequestException:
return []
```

Αρχικά πραγματοποιείται έλεγχος εγκυρότητας του όρου αναζήτησης. Εάν ο όρος είναι κενός, η συνάρτηση επιστρέφει αμέσως κενή λίστα χωρίς να πραγματοποιηθεί κλήση στο API. Στη συνέχεια ορίζεται η βασική διεύθυνση URL του Open Library API καθώς και οι παράμετροι του αιτήματος. Ανάλογα με το κριτήριο φιλτραρίσματος που επέλεξε ο χρήστης, προστίθεται η κατάλληλη παράμετρος στο αίτημα. Εάν το κριτήριο είναι 'title' ο όρος αναζήτησης αποστέλλεται στο πεδίο title, εάν είναι 'author' αποστέλλεται στο πεδίο author, εάν είναι 'genre' αποστέλλεται στο πεδίο subject, ενώ σε κάθε άλλη περίπτωση αποστέλλεται ως γενικός όρος στο πεδίο q. Το αίτημα αποστέλλεται με χρονικό όριο 8 δευτερολέπτων. Εάν η αποστολή αποτύχει λόγω υπέρβασης του χρονικού ορίου ή άλλου σφάλματος δικτύου, η συνάρτηση επιστρέφει κενή λίστα. Για κάθε βιβλίο που επιστρέφεται δημιουργείται ένα λεξικό Python

που περιέχει τον μοναδικό αναγνωριστικό (key), τον τίτλο (title), τον συγγραφέα (author) όπου εμφανίζονται μόνο οι δύο πρώτοι σε περίπτωση πολλαπλών συγγραφέων, το έτος πρώτης έκδοσης (year), τη διεύθυνση URL του εξωφύλλου (cover_url) η οποία κατασκευάζεται βάσει του αναγνωριστικού cover_i, τη μέση βαθμολογία (rating) στρογγυλοποιημένη σε ένα δεκαδικό ψηφίο, τον συνολικό αριθμό αξιολογήσεων (ratings_count) και τα λογοτεχνικά είδη (genres) όπου εμφανίζονται μόνο τα τρία πρώτα. Τέλος τα βιβλία ταξινομούνται βάσει δημοτικότητας μέσω της sort_by_popularity(). Αφού συγκεντρωθούν όλα τα βιβλία ακολουθεί η ταξινόμησή τους βάσει ενός δείκτη δημοτικότητας μέσω της συνάρτησης sort_by_popularity() που υλοποιείται στο αρχείο popularity.py. Ο πλήρης κώδικας του αρχείου αυτού παρατίθεται ακολούθως:

Κώδικας από το αρχείο popularity.py : Υπολογισμός δημοτικότητας:

```
import math
def popularity_score(rating, ratings_count):
 # Formula: score = rating * log(1 + ratings_count)
 # Παραδειγματα:
 # rating=5.0, count=2 -> 5.0 * log(3) = 5.49
 # rating=4.5, count=1000 -> 4.5 * log(1001) = 31.1 (καλυτερο)
 # rating=4.0, count=5000 -> 4.0 * log(5001) = 34.0 (ακομα καλυτερο)
 if not rating or not ratings_count:
 return 0.0
 return round(rating * math.log(1 + ratings_count), 4)
def sort_by_popularity(books):
 for book in books:
 book['popularity_score'] = popularity_score(
 book.get('rating', 0),
 book.get('ratings_count', 0)
 )
 return sorted(books, key=lambda b: b['popularity_score'], reverse=True)
def get_popularity_tier(score):
 if score >= 30:
 return 'Πολυ δημοφιλες'
 elif score >= 15:
 return 'Δημοφιλες'
 elif score >= 5:
 return 'Γνωστο'
 else:
 return ''
```

Η λογική πίσω από τον τύπο popularity_score = rating x log(1 + ratings_count) είναι η εξής: ένα βιβλίο με υψηλή βαθμολογία αλλά πολύ μικρό αριθμό αξιολογήσεων δεν είναι εξίσου αξιόπιστο με ένα βιβλίο που έχει ελαφρώς χαμηλότερη βαθμολογία αλλά σημαντικά μεγαλύτερο αριθμό αξιολογήσεων. Για παράδειγμα, ένα βιβλίο με rating 5.0 και μόλις 2 αξιολογήσεις έχει popularity score 5.49, ενώ ένα βιβλίο με rating 4.5 και 1000 αξιολογήσεις έχει score 31.1, το οποίο είναι σημαντικά υψηλότερο. Η χρήση της λογαριθμικής συνάρτησης log() εξασφαλίζει ότι βιβλία με εκατομμύρια αξιολογήσεων δεν κυριαρχούν πλήρως στα αποτελέσματα. Μέσω της συνάρτησης get_popularity_tier() αποδίδεται σε κάθε βιβλίο μια κατηγορία δημοτικότητας που εμφανίζεται ως διακριτικό στο γραφικό περιβάλλον. Βιβλία με

score >= 30 χαρακτηρίζονται ως 'Πολύ δημοφιλές', με score >= 15 ως 'Δημοφιλές' και με score >= 5 ως 'Γνωστό'. Ένα ιδιαίτερο χαρακτηριστικό της εφαρμογής BookLovers είναι ότι η καταγραφή γεγονότων (event logging) πραγματοποιείται ακόμα και για μη εγγεγραμμένους χρήστες. Κάθε φορά που ένας επισκέπτης πραγματοποιεί αναζήτηση ή κάνει κλικ σε κάποιο βιβλίο, το σύστημα καταγράφει αυτόματα το γεγονός αυτό στον πίνακα EVENT της βάσης δεδομένων. Ο πλήρης κώδικας της συνάρτησης log_event() καθώς και των διαδρομών καταγραφής γεγονότων παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py : Καταγραφή γεγονότων χρήστη:

```
def log_event(event_type, book_key, book_title='', weight=1.0):
 if 'session_id' not in session:
 session['session_id'] = str(uuid.uuid4())
 event = Event(
 user_id = current_user.id if current_user.is_authenticated else None,
 book_key = book_key,
 book_title = book_title,
 event_type = event_type,
 weight = weight,
 session_id = session['session_id'],
 timestamp = datetime.utcnow()
 )
 db.session.add(event)
 db.session.commit()
@app.route('/event/click', methods=['POST'])
def event_click():
 data = request.get_json()
 log_event(event_type='clicked', book_key=data.get('book_key', ''),
 book_title=data.get('book_title', ''), weight=2.0)
 return jsonify({'status': 'ok'})
@app.route('/event/wishlist', methods=['POST'])
def event_wishlist():
 data = request.get_json()
 log_event(event_type='wishlist', book_key=data.get('book_key', ''),
 book_title=data.get('book_title', ''), weight=3.0)
 return jsonify({'status': 'ok', 'message': 'Προστέθηκε στο Wishlist!'})
```

Η συνάρτηση log_event() αρχικά ελέγχει εάν υπάρχει ήδη αναγνωριστικό συνεδρίας (session_id) αποθηκευμένο στη συνεδρία του χρήστη. Εάν δεν υπάρχει, δημιουργείται ένα νέο μοναδικό αναγνωριστικό μέσω της συνάρτησης uuid.uuid4(), το οποίο χρησιμοποιείται για την ομαδοποίηση των γεγονότων που πραγματοποιήθηκαν κατά τη διάρκεια της ίδιας επίσκεψης. Για το πεδίο user_id ελέγχεται εάν ο τρέχων χρήστης είναι συνδεδεμένος μέσω της ιδιότητας current_user.is_authenticated. Εάν είναι συνδεδεμένος αποθηκεύεται το αναγνωριστικό του, ενώ σε αντίθετη περίπτωση το πεδίο λαμβάνει την τιμή None. Τέλος το αντικείμενο αποθηκεύεται στη βάση δεδομένων μέσω των εντολών db.session.add() και db.session.commit(). Τα βάρη που αποδίδονται στα διαφορετικά είδη γεγονότων διαφέρουν ανάλογα με το επίπεδο ενδιαφέροντος που εκφράζει κάθε ενέργεια. Μια αναζήτηση (searched) αποδίδει βάρος 2.0, ένα κλικ σε βιβλίο (clicked) αποδίδει επίσης βάρος 2.0, ενώ η προσθήκη βιβλίου στη λίστα επιθυμιών (wishlist) αποδίδει βάρος 3.0, καθώς θεωρείται ότι εκφράζει υψηλότερο επίπεδο ενδιαφέροντος. Η διαδρομή /event/click καταγράφει το κλικ του χρήστη

σε κάποιο βιβλίο, ενώ η διαδρομή /event/wishlist καταγράφει την προσθήκη βιβλίου στη λίστα επιθυμιών. Και οι δύο διαδρομές δέχονται αίτημα τύπου POST με δεδομένα JSON και επιστρέφουν αντίστοιχη απάντηση επιβεβαίωσης. Συνοψίζοντας, ο μη εγγεγραμμένος χρήστης έχει πρόσβαση στην αρχική σελίδα, στην αναζήτηση βιβλίων βάσει τίτλου, συγγραφέα ή λογοτεχνικού είδους, και στην προβολή αποτελεσμάτων με πληροφορίες όπως τίτλος, συγγραφέας, εξώφυλλο, βαθμολογία και λογοτεχνικό είδος. Αντίθετα, η προσθήκη βιβλίων στη λίστα επιθυμιών, η λήψη εξατομικευμένων προτάσεων και η πρόσβαση στο προσωπικό ταμπλό απαιτούν ενεργό λογαριασμό. Εάν ένας μη εγγεγραμμένος χρήστης επιχειρήσει να αποκτήσει πρόσβαση σε αυτές τις σελίδες, το σύστημα τον ανακατευθύνει αυτόματα στη σελίδα σύνδεσης μέσω του decorator @login_required, όπως φαίνεται στον ακόλουθο κώδικα:

Κομμάτι κώδικα από το αρχείο app.py : Προστασία σελίδων με

```
@app.route('/recommendations')
@login_required
def recommendations():
 events = Event.query.filter_by(user_id=current_user.id)\
 .order_by(Event.timestamp.desc())\
 .limit(50).all()
 recs = get_recommendations(events)
 stats = {
 'total': len(events),
 'searches': sum(1 for e in events if e.event_type == 'searched'),
 'clicks': sum(1 for e in events if e.event_type == 'clicked'),
 'wishlist': sum(1 for e in events if e.event_type == 'wishlist'),
 }
 return render_template('recommendations.html',
 recommendations=recs, stats=stats)
@app.route('/dashboard')
@login_required
def dashboard():
 return render_template('BookStoreMainPage.html', user=current_user)
@app.route('/logout')
@login_required
def logout():
 logout_user()
 flash('Αποσυνδεθήκατε.', 'info')
 return redirect(url_for('login'))
```

Η χρήση του @login_required πάνω από τον ορισμό των συναρτήσεων recommendations(), dashboard() και logout() εξασφαλίζει ότι οι σελίδες αυτές είναι προσβάσιμες μόνο από συνδεδεμένους χρήστες. Εάν ένας μη εγγεγραμμένος χρήστης επιχειρήσει να αποκτήσει πρόσβαση σε αυτές τις διευθύνσεις, η βιβλιοθήκη Flask-Login τον ανακατευθύνει αυτόματα στη σελίδα σύνδεσης που έχει οριστεί μέσω της ρύθμισης login_manager.login_view = 'login'. Με αυτόν τον τρόπο επιτυγχάνεται σαφής διαχωρισμός μεταξύ των λειτουργιών που είναι ελεύθερα προσβάσιμες και εκείνων που απαιτούν πιστοποίηση ταυτότητας, διασφαλίζοντας παράλληλα την ασφάλεια και την ακεραιότητα των δεδομένων της εφαρμογής

3.2 Εγγραφή και Σύνδεση Χρήστη

Η ενότητα αυτή αναλύει τη διαδικασία εγγραφής νέου χρήστη και σύνδεσης υπάρχοντος χρήστη στην εφαρμογή BookLovers. Πρόκειται για δύο από τις πιο κρίσιμες λειτουργίες της εφαρμογής, καθώς αποτελούν την πύλη εισόδου στις εξατομικευμένες δυνατότητές της. Η υλοποίησή τους βασίζεται στη βιβλιοθήκη Flask-Login για τη διαχείριση της ταυτοποίησης των χρηστών και στη βιβλιοθήκη Werkzeug για την ασφαλή αποθήκευση των κωδικών πρόσβασης. Επιπλέον, έχει υλοποιηθεί ένα ξεχωριστό module ελέγχου πολιτικής κωδικών πρόσβασης, το οποίο εξασφαλίζει ότι κάθε νέος κωδικός πληροί συγκεκριμένες απαιτήσεις ασφαλείας. Πριν αναλυθεί η διαδικασία εγγραφής και σύνδεσης, είναι απαραίτητο να παρουσιαστεί η δομή του μοντέλου χρήστη, το οποίο ορίζεται στο αρχείο modelsDB.py. Το μοντέλο αυτό αποτελεί τη βάση πάνω στην οποία στηρίζονται όλες οι λειτουργίες διαχείρισης χρηστών της εφαρμογής. Ο πλήρης κώδικας του μοντέλου παρατίθεται ακολούθως:

Κώδικας από το αρχείο modelsDB.py Ορισμός μοντέλου χρήστη:

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import datetime
db = SQLAlchemy()
class User(UserMixin, db.Model):
 id = db.Column(db.Integer, primary_key=True)
 username = db.Column(db.String(50), unique=True, nullable=False)
 password = db.Column(db.String(200), nullable=False)
 email = db.Column(db.String(150), unique=True, nullable=False)
 def get_id(self):
 return str(self.id)
 @property
 def is_active(self):
 return True
 @property
 def is_authenticated(self):
 return True
 @property
 def is_anonymous(self):
 return False
```

Η κλάση User κληρονομεί τόσο από την κλάση UserMixin της βιβλιοθήκης Flask-Login όσο και από την κλάση db.Model της βιβλιοθήκης Flask-SQLAlchemy. Η κληρονομιά από την UserMixin παρέχει αυτόματα στην κλάση User μια σειρά από ιδιότητες και μεθόδους που απαιτούνται από τη Flask-Login για τη διαχείριση της ταυτοποίησης, όπως οι ιδιότητες is_active, is_authenticated και is_anonymous. Στην προκειμένη περίπτωση οι ιδιότητες αυτές ορίζονται ρητά ως properties ώστε να παρέχουν τις κατάλληλες τιμές για κάθε χρήστη που είναι εγγεγραμμένος στο σύστημα. Ο πίνακας User αποτελείται από τέσσερα πεδία. Το πεδίο id είναι ακέραιος αριθμός και αποτελεί το πρωτεύον κλειδί του πίνακα, αναγνωρίζοντας μοναδικά κάθε χρήστη. Το πεδίο username αποθηκεύει το μοναδικό όνομα χρήστη με μέγιστο μήκος 50 χαρακτήρων, ενώ ο περιορισμός unique=True εξασφαλίζει ότι δεν μπορούν να υπάρχουν δύο χρήστες με το ίδιο όνομα. Το πεδίο password αποθηκεύει τον κατακερματισμένο κωδικό πρόσβασης με μέγιστο μήκος 200 χαρακτήρων, καθώς οι κατακερματισμένοι κωδικοί είναι σημαντικά μεγαλύτεροι από τους αρχικούς. Τέλος, το πεδίο email αποθηκεύει τη

μοναδική διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη. Πριν αναλυθεί η διαδικασία εγγραφής, κρίνεται σκόπιμο να παρουσιαστεί το module ελέγχου πολιτικής κωδικών πρόσβασης, το οποίο υλοποιείται στο ξεχωριστό αρχείο validate_password_policies.py. Το module αυτό αναλαμβάνει τον έλεγχο εγκυρότητας του κωδικού πρόσβασης που εισάγει ο χρήστης κατά την εγγραφή του, εξασφαλίζοντας ότι αυτός πληροί συγκεκριμένες απαιτήσεις ασφαλείας. Ο πλήρης κώδικας του module παρατίθεται ακολούθως:

Κώδικας από το αρχείο validate_password_policies.py : Έλεγχος πολιτικής κωδικού:

```
import re
def validate_password(password):
 errors = []
 if len(password) < 8:
 errors.append('Τουλαχιστον 8 χαρακτηρες.')
 if not re.search(r'[A-Z]', password):
 errors.append('Τουλαχιστον 1 κεφαλαιο γραμμα (A-Z).')
 if not re.search(r'[a-z]', password):
 errors.append('Τουλαχιστον 1 πεζο γραμμα (a-z).')
 if not re.search(r'[0-9]', password):
 errors.append('Τουλαχιστον 1 αριθμο (0-9).')
 if not re.search(r'[!@#$%^&*()_+\-=\[\]{};:\|,.<>\/?]', password):
 errors.append('Τουλαχιστον 1 ειδικο χαρακτηρα (!@#$% κλπ.).')
 return errors
```

Η συνάρτηση validate_password() δέχεται ως παράμετρο τον κωδικό πρόσβασης που εισήγαγε ο χρήστης και επιστρέφει μια λίστα με μηνύματα σφάλματος. Εάν η λίστα είναι κενή, ο κωδικός θεωρείται έγκυρος. Συγκεκριμένα, πραγματοποιούνται πέντε έλεγχοι. Πρώτον, ελέγχεται εάν ο κωδικός έχει μήκος τουλάχιστον 8 χαρακτήρων. Δεύτερον, ελέγχεται εάν περιέχει τουλάχιστον ένα κεφαλαίο γράμμα μέσω της κανονικής έκφρασης r'[A-Z]'. Τρίτον, ελέγχεται εάν περιέχει τουλάχιστον ένα πεζό γράμμα μέσω της κανονικής έκφρασης r'[a-z]'. Τέταρτον, ελέγχεται εάν περιέχει τουλάχιστον έναν αριθμό μέσω της κανονικής έκφρασης r'[0-9]'. Πέμπτον, ελέγχεται εάν περιέχει τουλάχιστον έναν ειδικό χαρακτήρα όπως !@#$% κλπ. Για κάθε έλεγχο που αποτυγχάνει προστίθεται το αντίστοιχο μήνυμα σφάλματος στη λίστα errors, η οποία επιστρέφεται στο τέλος της συνάρτησης. Η διαδικασία εγγραφής νέου χρήστη υλοποιείται μέσω της διαδρομής '/SignUp' στο αρχείο app.py. Η διαδρομή αυτή δέχεται αιτήματα τόσο τύπου GET όσο και τύπου POST. Στην περίπτωση αιτήματος GET αποδίδεται η φόρμα εγγραφής, ενώ στην περίπτωση αιτήματος POST επεξεργάζονται τα δεδομένα που υπέβαλε ο χρήστης. Ο πλήρης κώδικας της συνάρτησης εγγραφής παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Εγγραφή νέου χρήστη:

```
@app.route('/SignUp', methods=['GET', 'POST'])
def SignUp():
 if request.method == 'POST':
 username = request.form.get('username')
 password = request.form.get('password')
 confirm_password = request.form.get('confirm_password')
 email = request.form.get('email')
 # Ελεγχος πολιτικης κωδικου
```

```
policy_errors = validate_password(password)
if policy_errors:
for err in policy_errors:
flash(err, 'error')
return redirect(url_for('SignUp'))
# Ελεγχος ταυτοτητας κωδικων
if password != confirm_password:
flash('Οι κωδικοι δεν ταιριαζουν.', 'error')
return redirect(url_for('SignUp'))
# Ελεγχος μοναδικοτητας username
existing_user = User.query.filter_by(username=username).first()
if existing_user:
flash('Το ονομα χρηστη υπαρχει ηδη.', 'error')
return redirect(url_for('SignUp'))
# Ελεγχος μοναδικοτητας email
existing_email = User.query.filter_by(email=email).first()
if existing_email:
flash('Το email χρησιμοποιειται ηδη.', 'error')
return redirect(url_for('SignUp'))
# Δημιουργια νεου χρηστη με κατακερματισμενο κωδικο
hashed_pw = generate_password_hash(password)
user = User(username=username, password=hashed_pw, email=email)
db.session.add(user)
db.session.commit()
flash('Εγγραφηκατε με επιτυχια. Καντε εισοδο.', 'success')
return redirect(url_for('login'))
return render_template('SignUp.html')
```

Η συνάρτηση SignUp() υλοποιεί μια σειρά από ελέγχους πριν προχωρήσει στη δημιουργία του νέου λογαριασμού. Αρχικά ανακτώνται τα δεδομένα που υπέβαλε ο χρήστης μέσω της φόρμας εγγραφής, συγκεκριμένα το όνομα χρήστη (username), ο κωδικός πρόσβασης (password), η επιβεβαίωση κωδικού (confirm_password) και η διεύθυνση ηλεκτρονικού ταχυδρομείου (email). Στη συνέχεια πραγματοποιείται ο πρώτος έλεγχος μέσω της κλήσης της συνάρτησης validate_password(). Εάν ο κωδικός δεν πληροί τις απαιτήσεις πολιτικής, κάθε μήνυμα σφάλματος εμφανίζεται στον χρήστη μέσω της συνάρτησης flash() και ο χρήστης ανακατευθύνεται εκ νέου στη φόρμα εγγραφής. Εάν ο κωδικός είναι έγκυρος, πραγματοποιείται ο δεύτερος έλεγχος, ο οποίος αφορά την ταυτότητα των δύο κωδικών που εισήγαγε ο χρήστης. Εάν ο κωδικός και η επιβεβαίωσή του δεν συμφωνούν, εμφανίζεται κατάλληλο μήνυμα σφάλματος. Ακολουθεί ο τρίτος έλεγχος, ο οποίος αφορά τη μοναδικότητα του ονόματος χρήστη. Μέσω ερωτήματος στη βάση δεδομένων User.query.filter_by(username=username).first() ελέγχεται εάν υπάρχει ήδη χρήστης με το ίδιο όνομα. Εάν υπάρχει, εμφανίζεται αντίστοιχο μήνυμα σφάλματος. Αντίστοιχος έλεγχος πραγματοποιείται και για τη διεύθυνση ηλεκτρονικού ταχυδρομείου, εξασφαλίζοντας ότι κάθε email μπορεί να χρησιμοποιηθεί για έναν μόνο λογαριασμό. Εφόσον όλοι οι έλεγχοι ολοκληρωθούν με επιτυχία, προχωρά η διαδικασία δημιουργίας του νέου λογαριασμού. Ο κωδικός πρόσβασης κατακερματίζεται μέσω της συνάρτησης generate_password_hash() της βιβλιοθήκης Werkzeug πριν αποθηκευτεί στη βάση δεδομένων. Ο κατακερματισμός αυτός εξασφαλίζει ότι ακόμα και σε περίπτωση μη εξουσιοδοτημένης πρόσβασης στη βάση δεδομένων, οι κωδικοί των χρηστών παραμένουν προστατευμένοι, καθώς δεν είναι δυνατή η

ανάκτηση του αρχικού κωδικού από τον κατακερματισμένο. Στη συνέχεια δημιουργείται ένα νέο αντικείμενο User με τα στοιχεία του χρήστη, το οποίο αποθηκεύεται στη βάση δεδομένων μέσω των εντολών db.session.add() και db.session.commit(). Τέλος εμφανίζεται μήνυμα επιτυχίας και ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης. Η διαδικασία σύνδεσης υπάρχοντος χρήστη υλοποιείται μέσω της διαδρομής '/Login' στο αρχείο app.py. Όπως και η εγγραφή, η διαδρομή αυτή δέχεται αιτήματα τόσο τύπου GET όσο και τύπου POST. Στην περίπτωση αιτήματος GET αποδίδεται η φόρμα σύνδεσης, ενώ στην περίπτωση αιτήματος POST επεξεργάζονται τα διαπιστευτήρια που υπέβαλε ο χρήστης. Ο πλήρης κώδικας της συνάρτησης σύνδεσης παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Σύνδεση χρήστη:

```
@app.route('/Login', methods=['GET', 'POST'])
def login():
 image_url = url_for('static', filename='bookstore.jpg')
 if request.method == 'POST':
 username = request.form.get('username')
 password = request.form.get('password')
 # Αναζητηση χρηστη στη βαση δεδομενων
 user = User.query.filter_by(username=username).first()
 # Επαληθευση κωδικου
 if user and check_password_hash(user.password, password):
 login_user(user, remember=True)
 flash('Επιτυχης εισοδος!', 'success')
 return redirect(url_for('dashboard'))
 else:
 flash('Λαθος στοιχεια. Προσπαθηστε ξανα.', 'error')
 return render_template('Login.html', image_url=image_url)
```

Η συνάρτηση login() αρχικά ανακτά τη διεύθυνση URL της εικόνας φόντου της σελίδας σύνδεσης μέσω της url_for('static', filename='bookstore.jpg'), η οποία μεταφέρεται ως παράμετρος στο πρότυπο Login.html. Στη συνέχεια, εφόσον το αίτημα είναι τύπου POST, ανακτώνται το όνομα χρήστη και ο κωδικός πρόσβασης από τη φόρμα σύνδεσης. Ακολούθως πραγματοποιείται αναζήτηση στη βάση δεδομένων για τον χρήστη με το δεδομένο όνομα μέσω του ερωτήματος User.query.filter_by(username=username).first(). Εάν βρεθεί χρήστης με αυτό το όνομα, ελέγχεται ο κωδικός πρόσβασης μέσω της συνάρτησης check_password_hash() της βιβλιοθήκης Werkzeug. Η συνάρτηση αυτή συγκρίνει τον κωδικό που εισήγαγε ο χρήστης με τον κατακερματισμένο κωδικό που είναι αποθηκευμένος στη βάση δεδομένων, χωρίς να χρειάζεται να αποκρυπτογραφηθεί ο τελευταίος. Εάν τα διαπιστευτήρια είναι ορθά, καλείται η συνάρτηση login_user(user, remember=True) η οποία συνδέει τον χρήστη στο σύστημα και δημιουργεί cookie μακράς διάρκειας 7 ημερών λόγω της παραμέτρου remember=True. Σε αντίθετη περίπτωση εμφανίζεται μήνυμα σφάλματος και ο χρήστης καλείται να εισάγει εκ νέου τα στοιχεία του. Η διαδικασία αποσύνδεσης του χρήστη υλοποιείται μέσω της διαδρομής '/logout' στο αρχείο app.py. Η διαδρομή αυτή είναι προστατευμένη με τον decorator @login_required, εξασφαλίζοντας ότι μόνο συνδεδεμένοι χρήστες μπορούν να την καλέσουν. Ο κώδικας της συνάρτησης αποσύνδεσης παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Αποσύνδεση χρήστη:

```
@app.route('/logout')
```

```
@login_required
def logout():
 logout_user()
 flash('Αποσυνδεθηκατε.', 'info')
 return redirect(url_for('login'))
```

Η συνάρτηση logout() είναι ιδιαίτερα απλή στην υλοποίησή της. Καλεί τη συνάρτηση logout_user() της βιβλιοθήκης Flask-Login, η οποία αναλαμβάνει να διαγράψει τα στοιχεία ταυτοποίησης του χρήστη από τη συνεδρία και να ακυρώσει το cookie σύνδεσης. Στη συνέχεια εμφανίζεται ενημερωτικό μήνυμα αποσύνδεσης μέσω της flash() και ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης. Τέλος, αξίζει να αναφερθεί ο τρόπος με τον οποίο δημιουργούνται οι πίνακες της βάσης δεδομένων κατά την εκκίνηση της εφαρμογής. Στο κάτω μέρος του αρχείου app.py υπάρχει το ακόλουθο κομμάτι κώδικα:

Κώδικας από το αρχείο app.py Εκκίνηση εφαρμογής και δημιουργία βάσης:

```
if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)
```

Κατά την εκκίνηση της εφαρμογής, εκτελείται η εντολή db.create_all() εντός του context της εφαρμογής Flask. Η εντολή αυτή αναλαμβάνει να δημιουργήσει αυτόματα όλους τους πίνακες της βάσης δεδομένων που ορίζονται ως κλάσεις στο αρχείο modelsDB.py, εφόσον αυτοί δεν υπάρχουν ήδη. Συγκεκριμένα δημιουργούνται οι πίνακες User, Book, UserPreference και Event με όλα τα πεδία και τους περιορισμούς που ορίζονται στις αντίστοιχες κλάσεις. Εάν οι πίνακες υπάρχουν ήδη, η εντολή δεν πραγματοποιεί καμία αλλαγή, εξασφαλίζοντας έτσι ότι τα υπάρχοντα δεδομένα δεν διαγράφονται κατά τις επόμενες εκκινήσεις της εφαρμογής. Η παράμετρος debug=True ενεργοποιεί τη λειτουργία αποσφαλμάτωσης του Flask, η οποία παρέχει λεπτομερή μηνύματα σφάλματος και επιτρέπει την αυτόματη επανεκκίνηση του διακομιστή κατά τη διάρκεια της ανάπτυξης της εφαρμογής.

3.3 Η εφαρμογή από την πλευρά του εγγεγραμμένου χρήστη και εξατομικευμενες προτασεις

Η ενότητα αυτή αναλύει τις λειτουργίες που είναι διαθέσιμες αποκλειστικά στον εγγεγραμμένο χρήστη της εφαρμογής BookLovers. Μόλις ο χρήστης συνδεθεί επιτυχώς στο σύστημα, αποκτά πρόσβαση σε μια σειρά από εξατομικευμένες δυνατότητες που δεν είναι διαθέσιμες στον ανώνυμο επισκέπτη. Συγκεκριμένα, ο εγγεγραμμένος χρήστης μπορεί να πραγματοποιεί αναζήτηση βιβλίων με πλήρη καταγραφή των ενεργειών του, να προσθέτει βιβλία στη λίστα επιθυμιών του, να κάνει κλικ στις λεπτομέρειες κάθε βιβλίου και να λαμβάνει εξατομικευμένες προτάσεις βιβλίων που βασίζονται στο ιστορικό των ενεργειών του. Το σύνολο αυτών των λειτουργιών αποτελεί τον πυρήνα της εμπειρίας που προσφέρει η εφαρμογή BookLovers στον χρήστη της. Μετά την επιτυχή σύνδεση, ο χρήστης ανακατευθύνεται στο ταμπλό (dashboard) της εφαρμογής, το οποίο αντιστοιχεί στη διαδρομή '/dashboard'. Η διαδρομή αυτή είναι προστατευμένη με τον decorator @login_required, εξασφαλίζοντας ότι μόνο συνδεδεμένοι χρήστες μπορούν να την προσπελάσουν. Το ταμπλό αποδίδει το κεντρικό πρότυπο της εφαρμογής μεταφέροντας το αντικείμενο current_user ως παράμετρο, ώστε το γραφικό περιβάλλον να μπορεί να εμφανίσει εξατομικευμένες πληροφορίες για τον συνδεδεμένο χρήστη, όπως το όνομά του. Ο κώδικας της διαδρομής παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Ταμπλό εγγεγραμμένου χρήστη:

```
@app.route('/dashboard')
@login_required
def dashboard():
 return render_template('BookStoreMainPage.html', user=current_user)
```

Η χρήση της παραμέτρου user=current_user στην κλήση της render_template() επιτρέπει στο πρότυπο Jinja2 να αναγνωρίσει τον συνδεδεμένο χρήστη και να προσαρμόσει αναλόγως το περιεχόμενο της σελίδας. Συγκεκριμένα, το πρότυπο BookStoreMainPage.html μπορεί να εμφανίσει το όνομα του χρήστη στη μπάρα πλοήγησης, να ενεργοποιήσει επιπλέον επιλογές όπως ο σύνδεσμος προς τις προτάσεις και τη λίστα επιθυμιών, και να αποκρύψει τους συνδέσμους εγγραφής και σύνδεσης που εμφανίζονται στον ανώνυμο επισκέπτη. Ο εγγεγραμμένος χρήστης έχει πρόσβαση στην ίδια λειτουργία αναζήτησης βιβλίων που περιγράφηκε στην ενότητα 3.1, με τη σημαντική διαφορά ότι η καταγραφή των γεγονότων αναζήτησης συνδέεται πλέον με το αναγνωριστικό του συγκεκριμένου χρήστη. Αυτό σημαίνει ότι κάθε αναζήτηση που πραγματοποιεί ο εγγεγραμμένος χρήστης αποθηκεύεται στη βάση δεδομένων με το user_id του, επιτρέποντας στο σύστημα να διαμορφώσει ένα δυναμικό προφίλ προτιμήσεων που εξελίσσεται με κάθε νέα ενέργεια. Επιπλέον, ο εγγεγραμμένος χρήστης μπορεί να κάνει κλικ στο κουμπί 'Λεπτομέρειες' κάθε βιβλίου, ενέργεια που καταγράφεται ως γεγονός τύπου 'clicked' με βάρος 2.0, καθώς και να προσθέσει βιβλία στη λίστα επιθυμιών του μέσω του κουμπιού wishlist. Ο κώδικας που διαχειρίζεται αυτές τις ενέργειες παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Καταγραφή κλικ και wishlist:

```
# Καταγραφη κλικ σε βιβλιο (βαρος 2.0)
@app.route('/event/click', methods=['POST'])
def event_click():
 data = request.get_json()
 log_event(
 event_type = 'clicked',
 book_key = data.get('book_key', ''),
 book_title = data.get('book_title', ''),
 weight = 2.0
 )
 return jsonify({'status': 'ok'})
# Καταγραφη προσθηκης στο wishlist (βαρος 3.0)
@app.route('/event/wishlist', methods=['POST'])
def event_wishlist():
 data = request.get_json()
 log_event(
 event_type = 'wishlist',
 book_key = data.get('book_key', ''),
 book_title = data.get('book_title', ''),
 weight = 3.0
 )
 return jsonify({'status': 'ok', 'message': 'Προστεθηκε στο Wishlist!'})
# Κεντρικη συναρτηση καταγραφης γεγονοτων
def log_event(event_type, book_key, book_title='', weight=1.0):
 if 'session_id' not in session:
 session['session_id'] = str(uuid.uuid4())
 event = Event(
 user_id = current_user.id if current_user.is_authenticated else None,
 book_key = book_key,
 book_title = book_title,
 event_type = event_type,
 weight = weight,
 session_id = session['session_id'],
 timestamp = datetime.utcnow()
 )
 db.session.add(event)
 db.session.commit()
```

Η διαδρομή /event/click δέχεται αίτημα τύπου POST με δεδομένα JSON που περιέχουν το αναγνωριστικό (book_key) και τον τίτλο (book_title) του βιβλίου στο οποίο έκανε κλικ ο χρήστης. Το γεγονός καταγράφεται με τύπο 'clicked'και βάρος 2.0. Αντίστοιχα, η διαδρομή /event/wishlist καταγράφει την προσθήκη βιβλίου στη λίστα επιθυμιών με τύπο 'wishlist' και βάρος 3.0, τοοποίο είναι υψηλότερο καθώς η ενέργεια αυτή εκφράζει ισχυρότερο ενδιαφέρον για το συγκεκριμένο βιβλίο. Στην περίπτωση του εγγεγραμμένου χρήστη, η συνάρτηση log_event() συμπληρώνει το πεδίο user_id με το αναγνωριστικό του τρέχοντος χρήστη μέσω

της ιδιότητας current_user.id, συνδέοντας έτσι μόνιμα το γεγονός με τον συγκεκριμένο λογαριασμό.
Το σύστημα αξιοποιεί τα καταγεγραμμένα γεγονότα κάθε χρήστη προκειμένουνα εξάγει τα ενδιαφέροντά του μέσω της συνάρτησης get_user_interests() που υλοποιείται στο αρχείο recommendations.py. Η συνάρτηση αυτή αναλύει το σύνολο των γεγονότων και δημιουργεί ένα σταθμισμένο λεξικό όρων που αντικατοπτρίζει τις αναγνωστικές προτιμήσεις του χρήστη. Ο πλήρης κώδικαστης συνάρτησης παρατίθεται ακολούθως:

Κώδικας από το αρχείο recommendations.py — Εξαγωγή ενδιαφερόντων χρήστη:

```
EVENT_WEIGHTS = {
 'searched': 2.0,
 'clicked': 2.0,
 'wishlist': 3.0,
}
GENRE_KEYWORDS = [
 'fiction', 'fantasy', 'mystery', 'romance', 'thriller',
 'science fiction', 'horror', 'history', 'biography',
 'adventure', 'crime', 'detective', 'magic', 'dystopia'
]
def get_user_interests(events):
 term_scores = Counter()
 for event in events:
 weight = EVENT_WEIGHTS.get(event.event_type, 1.0)
 title = (event.book_title or '').strip()
 if not title:
 continue
 if event.event_type == 'searched':
 term = title.lower().replace('search:', '').strip()
 if term:
 term_scores[term] += weight
 elif event.event_type in ('clicked', 'wishlist'):
 words = title.split()
 short = ' '.join(words[:2]).lower() if len(words) >= 2 else title.lower(
)
 term_scores[short] += weight
 return term_scores
```

Η συνάρτηση get_user_interests() χρησιμοποιεί ένα αντικείμενο Counter για την αθροιστική καταμέτρηση των σταθμισμένων όρων. Για κάθε γεγονός ανακτάται το βάρος του από το λεξικό EVENT_WEIGHTS. Εάν το γεγονός είναιαναζήτηση (searched), εξάγεται ο όρος αναζήτησης αφαιρώντας το πρόθεμα'search:' και προστίθεται στο Counter με το αντίστοιχο βάρος. Εάν το γεγονόςείναι κλικ (clicked) ή προσθήκη στο wishlist, εξάγονται οι δύο πρώτες λέξεις του τίτλου του βιβλίου ως αναπαραστατικός όρος, καθώς αυτές συνήθως αρκούν για τον χαρακτηρισμό του βιβλίου. Επιπλέον, έχει οριστεί μια λίστα GENRE_KEYWORDS που περιέχει 14 δημοφιλή λογοτεχνικά είδη και χρησιμοποιείται για τον εντοπισμό αναζητήσεων που αφορούν συγκεκριμένο είδος, ώστε να παραχθούν και προτάσεις βάσει είδους.
Η σελίδα προτάσεων αντιστοιχεί στη διαδρομή '/recommendations' και είναι προστατευμένη με @login_required. Κατά την επίσκεψη στη σελίδα αυτή, το σύστημα ανακτά τα 50 πιο

πρόσφατα γεγονότα του χρήστη από τη βάση δεδομένων, παράγει εξατομικευμένες προτάσεις και υπολογίζει στατιστικά στοιχεία για τη δραστηριότητα του χρήστη. Ο πλήρης κώδικας της διαδρομής παρατίθεται ακολούθως:

Κώδικας από το αρχείο app.py Σελίδα προτάσεων βιβλίων:

```
@app.route('/recommendations')
@login_required
def recommendations():
 # Ανακτηση 50 πιο προσφατων γεγονοτων του χρηστη
 events = Event.query.filter_by(user_id=current_user.id)\
 .order_by(Event.timestamp.desc())\
 .limit(50).all()
 # Παραγωγη εξατομικευμενων προτασεων
 recs = get_recommendations(events)
 # Υπολογισμος στατιστικων δραστηριοτητας χρηστη
 stats = {
 'total': len(events),
 'searches': sum(1 for e in events if e.event_type == 'searched'),
 'clicks': sum(1 for e in events if e.event_type == 'clicked'),
 'wishlist': sum(1 for e in events if e.event_type == 'wishlist'),
 }
 return render_template('recommendations.html',
 recommendations=recs, stats=stats)
```

Αρχικά πραγματοποιείται ερώτημα στη βάση δεδομένων για την ανάκτηση τωνγεγονότων του συγκεκριμένου χρήστη. Το ερώτημα φιλτράρει τα γεγονότα βάσει του user_id του τρέχοντος χρήστη, τα ταξινομεί κατά φθίνουσα χρονολογική σειρά και περιορίζει τα αποτελέσματα στα 50 πιο πρόσφατα. Ο περιορισμός αυτός εξασφαλίζει ότι το σύστημα προτάσεων βασίζεται στις πιο πρόσφατες και επομένως πιο σχετικές ενέργειες του χρήστη. Στη συνέχεια τα γεγονότα αυτά μεταφέρονται στη συνάρτηση get_recommendations() η οποία αναλαμβάνει την παραγωγή των προτάσεων. Παράλληλα υπολογίζονται στατιστικά στοιχεία για τη δραστηριότητα του χρήστη, όπως ο συνολικός αριθμός γεγονότων και η κατανομή τους ανά τύπο, τα οποία εμφανίζονται στησελίδα προτάσεων ενημερώνοντας τον χρήστη για το επίπεδο εξατομίκευσης που έχει επιτευχθεί.
Η κεντρική συνάρτηση παραγωγής προτάσεων είναι η get_recommendations()
που υλοποιείται στο αρχείο recommendations.py. Η συνάρτηση αυτή ενορχηστρώνει το σύνολο της διαδικασίας παραγωγής προτάσεων, συνδυάζοντας την ανάλυση γεγονότων, την ανάκτηση υποψήφιων βιβλίων απότο Open Library API, την εφαρμογή content-based filtering και τον υπολογισμό του υβριδικού βαθμού. Ο πλήρης κώδικας της συνάρτησης παρατίθεται ακολούθως:

Κώδικα από το αρχείο recommendations.py Συνάρτηση παραγωγής προτάσεων:

```
def get_recommendations(events):
 if not events:
 return []
 # Βημα 1: Εξαγωγη ενδιαφεροντων χρηστη
 term_scores = get_user_interests(events)
 if not term_scores:
 return []
 # Βημα 2: Επιλογη 4 κορυφαιων ορων
 top_terms = [term for term, _ in term_scores.most_common(4)]
 recommendations = []
 seen_keys = set()
 for term in top_terms:
 # Βημα 3: Ανακτηση υποψηφιων βιβλιων απο Open Library API
 books = fetch_books_for_term(term, limit=8)
 if not books:
 continue
 # Βημα 4: Εφαρμογη content-based filtering (TF-IDF)
 books = get_content_based_recommendations(events, books)
 # Βημα 5: Υπολογισμος υβριδικου βαθμου
 books = hybrid_score(books)
 # Βημα 6: Αφαιρεση διπλοτυπων
 unique = [b for b in books if b['key'] not in seen_keys]
 if unique:
 for b in unique:
 seen_keys.add(b['key'])
 recommendations.append({
 'reason': f'Επειδη αναζητησες "{term.title()}"',
 'type': 'search',
 'books': unique[:4]
 })
 # Βημα 7: Προτασεις βασει λογοτεχνικου ειδους
 matched_genre = next(
 (gk for gk in GENRE_KEYWORDS if gk in term.lower()), None
 )
 if matched_genre:
 genre_books = fetch_books_for_term(f'subject:{matched_genre}', limit=8)
 if genre_books:
 genre_books = get_content_based_recommendations(events, genre_books)
 genre_books = hybrid_score(genre_books)
 unique_genre = [b for b in genre_books if b['key'] not in seen_keys]
 if unique_genre:
 for b in unique_genre:
 seen_keys.add(b['key'])
 recommendations.append({
 'reason': f'Δημοφιλη στο ειδος "{matched_genre.title()}"',
 'type': 'genre',
```

```
'books': unique_genre[:4]
})
if len(recommendations) >= 4:
break
return recommendations
```

Η συνάρτηση get_recommendations() ακολουθεί μια σαφή αλληλουχία επτά βημάτων. Στο πρώτο βήμα ελέγχεται εάν υπάρχουν γεγονότα για επεξεργασία. Εάν η λίστα γεγονότων είναι κενή, επιστρέφεται αμέσως κενή λίστα. Στο δεύτερο βήμα καλείται η get_user_interests() για την εξαγωγή των σταθμισμένων ενδιαφερόντων του χρήστη. Στο τρίτο βήμα επιλέγονται οι 4 κορυφαίοι όροι βάσει σταθμισμένης συχνότητας εμφάνισης χρησιμοποιώντας τη μέθοδο most_common() του Counter. Για κάθε έναν από τους κορυφαίους όρους ακολουθούν τα υπόλοιπα βήματα. Στο τέταρτο βήμα ανακτώνται 8 υποψήφια βιβλία από το Open Library API μέσω της fetch_books_for_term(). Στο πέμπτο βήμα εφαρμόζεται content-based filtering μέσω TF-IDF και cosine similarity για τον υπολογισμό της ομοιότητας κάθε βιβλίου με το προφίλ του χρήστη. Στο έκτο βήμα υπολογίζεται ουβριδικός βαθμός που συνδυάζει TF-IDF και δημοτικότητα. Στο έβδομο βήμααφαιρούνται τα διπλότυπα μέσω του συνόλου seen_keys και τα μοναδικά βιβλία προστίθενται στη λίστα προτάσεων με αιτιολόγηση. Επιπλέον, εάν ο τρέχων όρος αναγνωριστεί ως λογοτεχνικό είδος βάσει της λίστας GENRE_KEYWORDS, παράγεται και μια επιπλέον ομάδα προτάσεων αποκλειστικά για το συγκεκριμένο είδος. Η διαδικασία σταματά όταν συγκεντρωθούν 4 ομάδες προτάσεων. Η ανάκτηση των υποψήφιων βιβλίων από το Open Library API για κάθε όροενδιαφέροντος υλοποιείται μέσω της συνάρτησης fetch_books_for_term() στο αρχείο recommendations.py. Η συνάρτηση αυτή διαφέρει από την search_books_openlibrary() του αρχείου search_api.py ως προς το ότι χρησιμοποιείται αποκλειστικά για εσωτερικές ανάγκες του συστήματος προτάσεων και όχι για την αναζήτηση που πραγματοποιεί ο χρήστης. Ο πλήρης κώδικας της συνάρτησης παρατίθεται ακολούθως:

Κώδικας από το αρχείο recommendations.py Ανάκτηση βιβλίων για έναν όρο:

```
def fetch_books_for_term(term, limit=8):
try:
response = requests.get(
'https://openlibrary.org/search.json',
params={
'q': term,
'limit': limit,
'fields': 'key,title,author_name,first_publish_year,
cover_i,subject,ratings_average,ratings_count'
},
timeout=8
)
response.raise_for_status()
data = response.json()
books = []
for doc in data.get('docs', []):
cover_id = doc.get('cover_i')
cover_url = (f'https://covers.openlibrary.org/b/id/{cover_id}-M.jpg'
if cover_id else None)
rating = doc.get('ratings_average', 0)
```

```
 rating = round(rating, 1) if rating else None
 subjects = doc.get('subject', [])
 books.append({
 'key': doc.get('key', ''),
 'title': doc.get('title', 'Unknown Title'),
 'author': ', '.join(doc.get('author_name', ['Unknown'])[:2]),
 'year': doc.get('first_publish_year', ''),
 'cover_url': cover_url,
 'rating': rating,
 'ratings_count': doc.get('ratings_count', 0),
 'genres': subjects[:3] if subjects else [],
 'ol_url': f'https://openlibrary.org{doc.get("key", "")}',
 })
 return books
 except Exception:
 return []
```

Η συνάρτηση fetch_books_for_term() αποστέλλει αίτημα GET στο Open Library API με τον δεδομένο όρο αναζήτησης και ανακτά έως 8 βιβλία. Για κάθε βιβλίο δημιουργείται ένα λεξικό με τα απαραίτητα πεδία: μοναδικό αναγνωριστικό, τίτλο, συγγραφέα, έτος έκδοσης, URL εξωφύλλου, βαθμολογία, αριθμό αξιολογήσεων, λογοτεχνικά είδη και σύνδεσμο προς το Open Library.Σε περίπτωση οποιουδήποτε σφάλματος κατά την επικοινωνία με το API, η συνάρτηση επιστρέφει κενή λίστα χωρίς να διακόπτεται η λειτουργία της εφαρμογής. Το content-based filtering υλοποιείται στο αρχείο content_based.py μέσω τριών συναρτήσεων: build_book_profile(), build_user_profile() και compute_tfidf_similarity(). Η κεντρική συνάρτηση get_content_based_recommendations() ενορχηστρώνει τη διαδικασία καλώντας τις παραπάνω συναρτήσεις. Ο πλήρης κώδικας του αρχείου παρατίθεται ακολούθως:

Κώδικας από το αρχείο content_based.py Content-based filtering με TF-IDF:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
def build_book_profile(book):
 # Δημιουργια text profile για καθε βιβλιο
 # Παραδειγμα: 'Harry Potter J.K. Rowling fantasy fiction magic'
 parts = []
 title = book.get('title', '')
 author = book.get('author', '')
 genres = book.get('genres', [])
 if title: parts.append(title)
 if author: parts.append(author)
 if genres: parts.extend(genres[:3])
 return ' '.join(parts).lower()
def build_user_profile(events):
 # Δημιουργια text profile χρηστη απο τα events του
 # Τα wishlist events εχουν διπλο βαρος
 parts = []
 for event in events:
```

```
 title = (event.book_title or '').strip()
 if not title: continue
 if event.event_type == 'wishlist':
 parts.append(title) # διπλο βαρος
 parts.append(title)
 elif event.event_type in ('clicked', 'searched'):
 parts.append(title)
 return ' '.join(parts).lower()
def compute_tfidf_similarity(user_profile_text, candidate_books):
 if not user_profile_text or not candidate_books:
 return candidate_books
 # Corpus: [0] = user profile, [1..N] = βιβλια
 book_profiles = [build_book_profile(b) for b in candidate_books]
 corpus = [user_profile_text] + book_profiles
 vectorizer = TfidfVectorizer(
 stop_words = 'english', # αφαιρει κοινες λεξεις (the, and, is...)
 ngram_range = (1, 2), # μονογραμμα και διγραμματα
 min_df = 1
 )
 try:
 tfidf_matrix = vectorizer.fit_transform(corpus)
 except Exception:
 return candidate_books
 # Cosine similarity: χρηστης [0] vs καθε βιβλιο [1..N]
 user_vector = tfidf_matrix[0]
 book_vectors = tfidf_matrix[1:]
 similarities = cosine_similarity(user_vector, book_vectors)[0]
 for i, book in enumerate(candidate_books):
 book['tfidf_score'] = round(float(similarities[i]), 4)
 return sorted(candidate_books, key=lambda b: b['tfidf_score'], reverse=True)
def get_content_based_recommendations(events, candidate_books):
 user_profile = build_user_profile(events)
 if not user_profile:
 return candidate_books
 return compute_tfidf_similarity(user_profile, candidate_books)
```

Η συνάρτηση build_book_profile() δημιουργεί ένα κείμενο-προφίλ για κάθε βιβλίο συνδυάζοντας τον τίτλο, τον συγγραφέα και τα λογοτεχνικά είδη σε μία ενιαία συμβολοσειρά πεζών χαρακτήρων. Αντίστοιχα, η build_user_profile() δημιουργεί το κείμενο-προφίλ του χρήστη από τα γεγονότα του, με τη σημαντική διαφορά ότι τα βιβλία που έχουν προστεθεί στο wishlist συμπεριλαμβάνονται δύο φορές, αποδίδοντάς τους έτσι διπλό βάρος στην αναπαράσταση TF-IDF.

Η συνάρτηση compute_tfidf_similarity() δημιουργεί το corpus κειμένων που αποτελείται από το προφίλ του χρήστη ως πρώτο στοιχείο και τα προφίλ των υποψήφιων βιβλίων στις υπόλοιπες θέσεις. Στη συνέχεια εφαρμόζεται TfidfVectorizer με αφαίρεση αγγλικών stop words και χρήση μονογραμμάτων και διγραμμάτων (ngram_range=(1,2)), ώστε να λαμβάνονται υπόψιν καιφράσεις δύο λέξεων. Το αποτέλεσμα είναι ένας πίνακας TF-IDF διανυσμάτων, στον οποίο κάθε γραμμή αναπαριστά ένα κείμενο του corpus. Τέλος υπολογίζεται η cosine similarity μεταξύ του διανύσματος του χρήστη (θέση 0) και των

διανυσμάτων κάθε βιβλίου (θέσεις 1 έως N), και το αποτέλεσμα αποθηκεύεται ως tfidf_score σε κάθε βιβλίο. Τα βιβλία ταξινομούνται κατά φθίνουσα σειρά tfidf_score πριν επιστραφούν. Το τελικό στάδιο του συστήματος προτάσεων είναι ο υπολογισμός του υβριδικού βαθμού, ο οποίος υλοποιείται στο αρχείο hybrid.py. Ο υβριδικός βαθμός συνδυάζει τον βαθμό TF-IDF ομοιότητας με τον βαθμό δημοτικότητας, εξασφαλίζοντας ότι οι προτάσεις που παράγονται είναι ταυτόχρονα σχετικές με τις προτιμήσεις του χρήστη και δημοφιλείς. Ο πλήρης κώδικας του αρχείου παρατίθεται ακολούθως:

Κώδικα από το αρχείο hybrid.py Υπολογισμός υβριδικού βαθμού:

```
from popularity import popularity_score
CONTENT_WEIGHT = 0.60 # TF-IDF similarity
POPULARITY_WEIGHT = 0.40 # popularity score (normalized)
def normalize(values):
 # Κανονικοποιηση τιμων στο [0, 1]
 # Αναγκαιο ωστε TF-IDF (0-1) και popularity (0-35+) να ειναι συγκρισιμα
 if not values:
 return values
 min_v = min(values)
 max_v = max(values)
 if max_v == min_v:
 return [0.5] * len(values)
 return [(v - min_v) / (max_v - min_v) for v in values]
def hybrid_score(books):
 # Τελικος τυπος: final_score = 0.60 * tfidf + 0.40 * popularity_normalized
 if not books:
 return books
 tfidf_scores = [b.get('tfidf_score', 0.0) for b in books]
 popularity_scores = [
 popularity_score(b.get('rating', 0), b.get('ratings_count', 0))
 for b in books
 ]
 norm_tfidf = normalize(tfidf_scores)
 norm_popularity = normalize(popularity_scores)
 for i, book in enumerate(books):
 book['popularity_score'] = round(popularity_scores[i], 4)
 book['norm_tfidf'] = round(norm_tfidf[i], 4)
 book['norm_popularity'] = round(norm_popularity[i], 4)
 book['hybrid_score'] = round(
 CONTENT_WEIGHT * norm_tfidf[i] +
 POPULARITY_WEIGHT * norm_popularity[i],
 4
 )
 return sorted(books, key=lambda b: b['hybrid_score'], reverse=True)
```

Ο υβριδικός αλγόριθμος ακολουθεί τα εξής βήματα. Αρχικά εξάγονται οι τιμές tfidf_score και popularity_score για κάθε βιβλίο. Ο βαθμός δημοτικότητας

υπολογίζεται μέσω της συνάρτησης popularity_score() που αναλύθηκε στην ενότητα 3.1. Στη συνέχεια και οι δύο τύποι βαθμών κανονικοποιούνται στο διάστημα [0,1] μέσω της συνάρτησης normalize(). Η κανονικοποίηση αυτή είναι απαραίτητη διότι ο βαθμός TF-IDF κυμαίνεται ήδη στο [0,1], ενώ ο βαθμός δημοτικότητας μπορεί να φτάσει τιμές άνω των 35, με αποτέλεσμα χωρίς κανονικοποίηση να κυριαρχεί αδικαιολόγητα στον τελικό βαθμό. Τέλος υπολογίζεται ο υβριδικός βαθμός ως σταθμισμένο άθροισμα των δύο κανονικοποιημένων βαθμών με αναλογία 60% TF-IDF και 40% δημοτικότητα. Η επιλογή της αναλογίας αυτής αντικατοπτρίζει την προτεραιότητα που δίνει η εφαρμογή στην εξατομίκευση έναντι της απλής δημοτικότητας, εξασφαλίζοντας παράλληλα ότι τα αποτελέσματα δεν θα αποκλίνουν από τις γενικά αναγνωρισμένες επιλογές. Τα βιβλία ταξινομούνται κατά φθίνουσα σειρά hybrid_score και επιστρέφονται ως τελικές προτάσεις προς τον χρήστη

Κεφάλαιο 4ο

4 Εγχειρίδιο χρήσης της εφαρμογής

Στο κεφάλαιο αυτό του παρόντος επιστημονικού εγγράφου παρατίθεται το εγχειρίδιο χρήσης της διαδικτυακής εφαρμογής BookLovers. Το εγχειρίδιο χρήσης κρίνεται απαραίτητο στην ανάλυση της εφαρμογής, καθώς παρέχει αναλυτική περιγραφή με βήματα ώστε όλοι οι χρήστες να έχουν τη δυνατότητα να χρησιμοποιήσουν την εφαρμογή αποτελεσματικά, ακόμη και στην περίπτωση που δεν έχουν χρησιμοποιήσει παρόμοια πλατφόρμα στο παρελθόν. Το γραφικό περιβάλλον της εφαρμογής έχει σχεδιαστεί με γνώμονα την απλότητα και την ευχρηστία, συντελώντας στη διαμόρφωση ενός φιλικού και οικείου πλαισίου για όλους τους υποψήφιους χρήστες. Στο παρόν κεφάλαιο περιγράφονται αναλυτικά δύο ξεχωριστά εγχειρίδια χρήσης, τα οποία αντιστοιχούν στους δύο διαφορετικούς τύπους χρηστών που υφίστανται στην εφαρμογή: τον μη εγγεγραμμένο και τον εγγεγραμμένο χρήστη. Επιπλέον, παρατίθενται τα βήματα εγκατάστασης της εφαρμογής, ώστε να είναι δυνατή η επιτυχής εκκίνηση και χρήση της σε οποιοδήποτε σύστημα.

4.1 Εγκατάσταση της Εφαρμογής

Για τη σωστή εγκατάσταση και χρήση της εφαρμογής, ο χρήστης θα πρέπει να διαθέτει εγκατεστημένο το πρόγραμμα PyCharm, από το οποίο θα ανοίξει την εφαρμογή. Μεταξύ των παραδοτέων της παρούσας εργασίας περιλαμβάνεται ο πλήρης φάκελος του project, ο οποίος περιέχει όλα τα αρχεία της εφαρμογής, συμπεριλαμβανομένου του αρχείου βάσης δεδομένων BookLovers.db που βρίσκεται στον υποφάκελο instance. Το αρχείο αυτό περιέχει τη δομή της βάσης δεδομένων καθώς και τα δεδομένα που έχουν δημιουργηθεί κατά τη χρήση της εφαρμογής, χωρίς να απαιτείται κάποιο επιπλέον πρόγραμμα διαχείρισης βάσης δεδομένων, καθώς η SQLite λειτουργεί αυτόνομα χωρίς την ανάγκη εξωτερικού διακομιστή.

Για την εγκατάσταση των απαραίτητων βιβλιοθηκών, ο χρήστης θα χρειαστεί να ανοίξει το Terminal του PyCharm και να εκτελέσει διαδοχικά τις ακόλουθες εντολές:
pip install flask
pip install flask-sqlalchemy
pip install flask-login
pip install werkzeug
pip install scikit-learn
pip install requests
pip install numpy

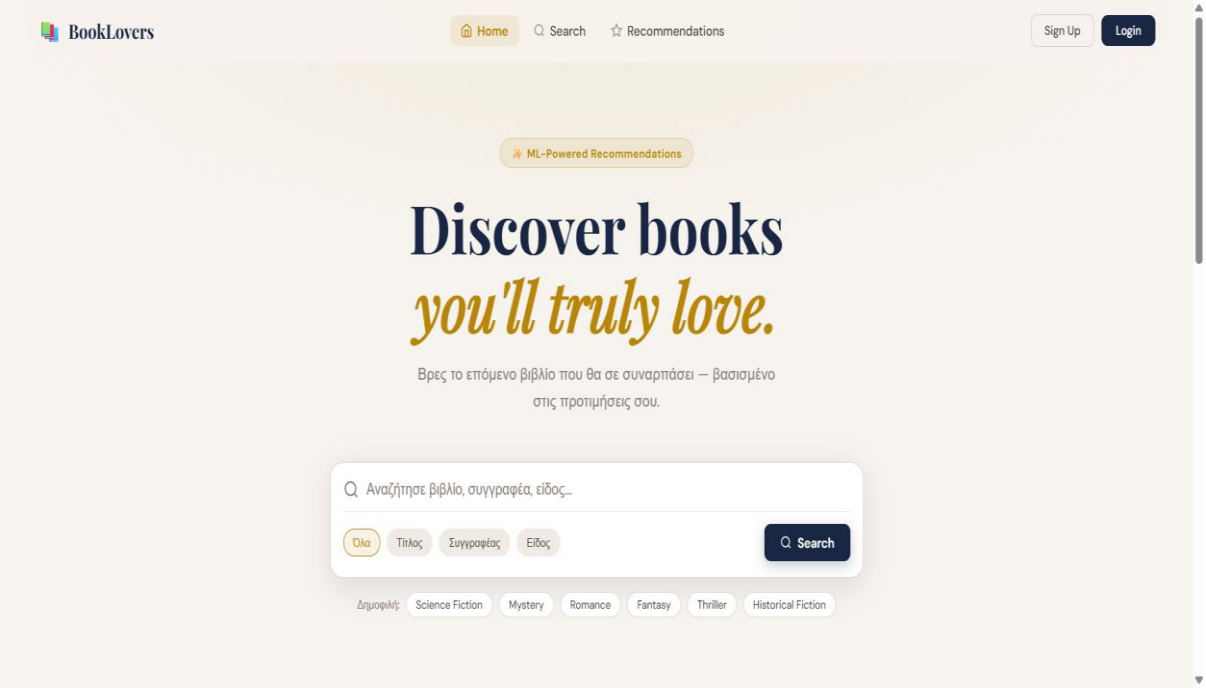
Μετά την ολοκλήρωση της εγκατάστασης των βιβλιοθηκών, ο χρήστης ανοίγει το αρχείο app.py από τον πλευρικό πίνακα του PyCharm και πατά το πράσινο κουμπί εκτέλεσης (Run). Μετά την επιτυχή εκκίνηση, εμφανίζεται στο παράθυρο Run η διεύθυνση http://127.0.0.1:5000 η ακόμα και http://booklovers.com:5000/ , την οποία ο χρήστης πληκτρολογεί στον περιηγητή ιστού της επιλογής του για να αποκτήσει πρόσβαση στην εφαρμογή BookLovers και να μπορέσει να συνδεθεί στο πρόγραμμα περιήγησης βιβλίων που επιθυμεί.

4.2 Εγχειρίδιο μη εγγεγραμμένου χρήστη

Όπως αναφέρθηκε αναλυτικά στα προηγούμενα κεφάλαια της παρούσας εργασίας, η εφαρμογή BookLovers απευθύνεται σε δύο διαφορετικά είδη χρηστών: τους εγγεγραμμένους και τους μη εγγεγραμμένους χρήστες. Οι δύο αυτές κατηγορίες χρηστών διαφέρουν ως προς τις δυνατότητες και τις λειτουργίες στις οποίες έχουν πρόσβαση, με τον εγγεγραμμένο χρήστη να απολαμβάνει ένα πλουσιότερο και πιο εξατομικευμένο περιβάλλον χρήσης. Στο παρόν υποκεφάλαιο παρουσιάζεται το εγχειρίδιο χρήσης της εφαρμογής από την πλευρά του μη εγγεγραμμένου χρήστη, δηλαδή του επισκέπτη που δεν διαθέτει λογαριασμό στην πλατφόρμα ή δεν έχει πραγματοποιήσει σύνδεση.

Το εγχειρίδιο αυτό απευθύνεται σε κάθε χρήστη που επισκέπτεται για πρώτη φορά την εφαρμογή BookLovers και επιθυμεί να εξερευνήσει τις βασικές δυνατότητές της χωρίς να απαιτείται η δημιουργία λογαριασμού. Στόχος του παρόντος εγχειριδίου είναι να καθοδηγήσει τον χρήστη βήμα προς βήμα στη χρήση της εφαρμογής, ώστε να μπορεί να αξιοποιήσει πλήρως τις διαθέσιμες λειτουργίες της με τον πιο απλό και αποτελεσματικό τρόπο.

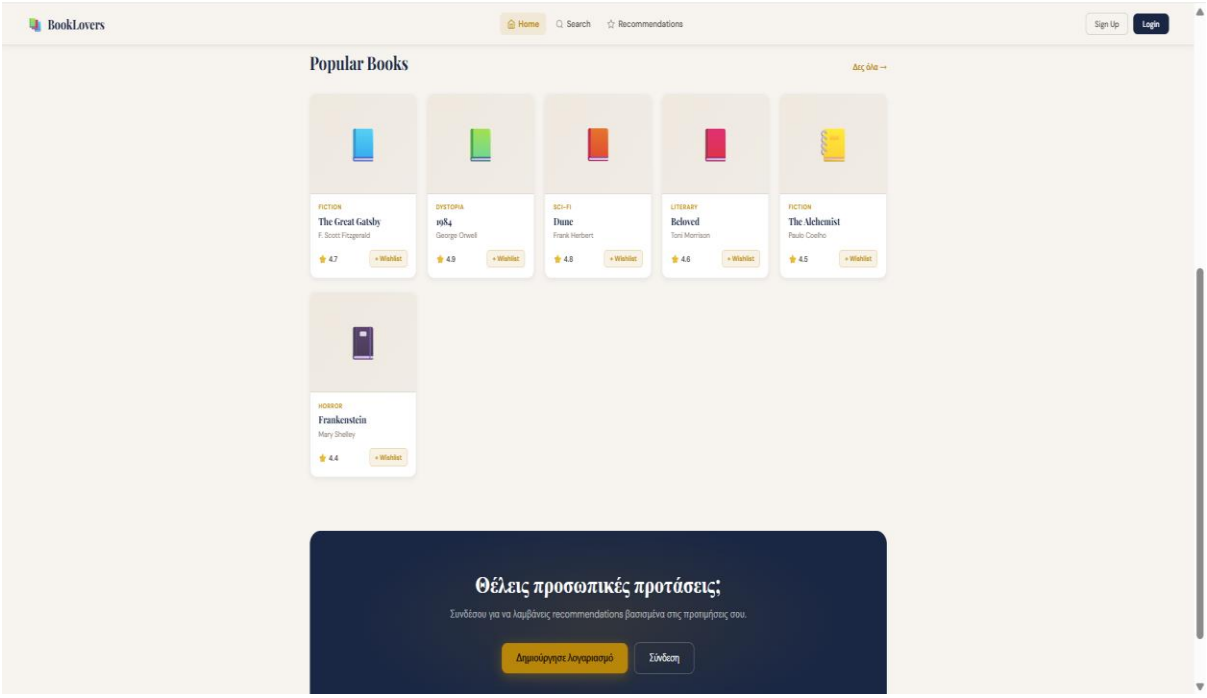
Αρχικά, μόλις ο χρήστης ανοίξει τη διαδικτυακή εφαρμογή BookLovers πληκτρολογώντας στον περιηγητή ιστού της επιλογής του τη διεύθυνση http://127.0.0.1:5000/, αντικρίζει την αρχική σελίδα της εφαρμογής. Η σελίδα αυτή αποτελεί το κεντρικό σημείο εισόδου στην πλατφόρμα και είναι σχεδιασμένη με τρόπο που να υποδέχεται τον επισκέπτη με φιλικό και κατανοητό τρόπο, παρέχοντάς του άμεση πρόσβαση στις βασικές λειτουργίες της εφαρμογής.



Εικόνα 5 Αρχική σελίδα εισόδου όλων των χρηστών (μέρος 1/2)

Στο πρώτο κομμάτι της αρχικής σελίδας μας όπως φαίνεται η εφαρμογή BookLovers αποτελείται από μια σαφή και φιλική διεπαφή προς τον χρήστη. Στο επάνω μέρος της σελίδας βρίσκεται η μπάρα πλοήγησης, η οποία περιλαμβάνει το λογότυπο της εφαρμογής στα αριστερά, καθώς και τρεις συνδέσμους πλοήγησης: Home, Search και Recommendations. Στη

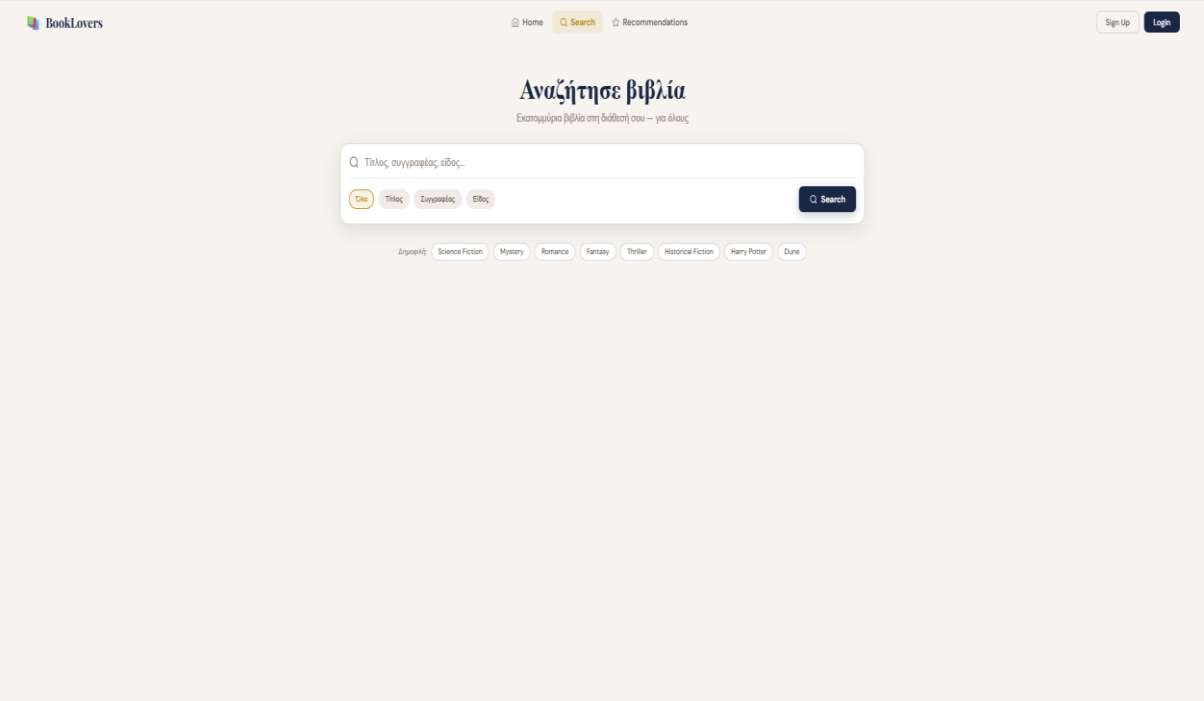
δεξιά πλευρά της μπάρας εμφανίζονται τα κουμπιά Sign Up και Login, τα οποία καθοδηγούν τον επισκέπτη προς τις σελίδες εγγραφής και σύνδεσης αντίστοιχα.
Αξίζει να σημειωθεί ότι ο σύνδεσμος Recommendations, αν και εμφανίζεται στη μπάρα πλοήγησης, δεν είναι προσβάσιμος από τον μη εγγεγραμμένο χρήστη. Εάν ο επισκέπτης επιχειρήσει να τον πατήσει, το σύστημα τον ανακατευθύνει αυτόματα στη σελίδα σύνδεσης, καθώς η λειτουργία των εξατομικευμένων προτάσεων απαιτεί την ύπαρξη ενεργού λογαριασμού.
Στο κεντρικό τμήμα της σελίδας εμφανίζεται το κύριο μήνυμα της εφαρμογής «Discover books you'll truly love», συνοδευόμενο από μια σύντομη περιγραφή που προσκαλεί τον χρήστη να ανακαλύψει το επόμενο βιβλίο του βασισμένο στις προτιμήσεις του. Ακριβώς από κάτω βρίσκεται το κεντρικό πεδίο αναζήτησης, το οποίο επιτρέπει στον χρήστη να αναζητήσει βιβλία πληκτρολογώντας τίτλο, συγγραφέα ή είδος. Το πεδίο αυτό συνοδεύεται από τέσσερα κουμπιά φιλτραρίσματος: Όλα, Τίτλος, Συγγραφέας και Είδος, τα οποία επιτρέπουν στον χρήστη να εξειδικεύσει την αναζήτησή του. Τέλος, στο κάτω μέρος της αναζήτησης εμφανίζονται δημοφιλή λογοτεχνικά είδη όπως Science Fiction, Mystery, Romance, Fantasy, Thriller και Historical Fiction, τα οποία λειτουργούν ως γρήγορες συντομεύσεις αναζήτησης για τον χρήστη.



Εικόνα 6 Αρχική σελίδα εισόδου όλων των χρηστών (μέρος 2/2)

Κάνοντας κύλιση προς τα κάτω στην αρχική σελίδα, ο μη εγγεγραμμένος χρήστης αντικρίζει την επόμενη οθόνη. Το δεύτερο αυτό τμήμα της αρχικής σελίδας αποτελείται από δύο ξεχωριστές περιοχές.Στο πρώτο τμήμα εμφανίζεται η ενότητα «Popular Books», η οποία παρουσιάζει μια συλλογή από δημοφιλή βιβλία που έχουν επιλεγεί βάσει του δείκτη δημοτικότητας της εφαρμογής. Κάθε βιβλίο εμφανίζεται σε ξεχωριστή κάρτα που περιλαμβάνει το εξώφυλλο, το λογοτεχνικό είδος, τον τίτλο, τον συγγραφέα, τη μέση βαθμολογία και το κουμπί «+ Wishlist». Συγκεκριμένα, στην οθόνη αυτή εμφανίζονται έξι

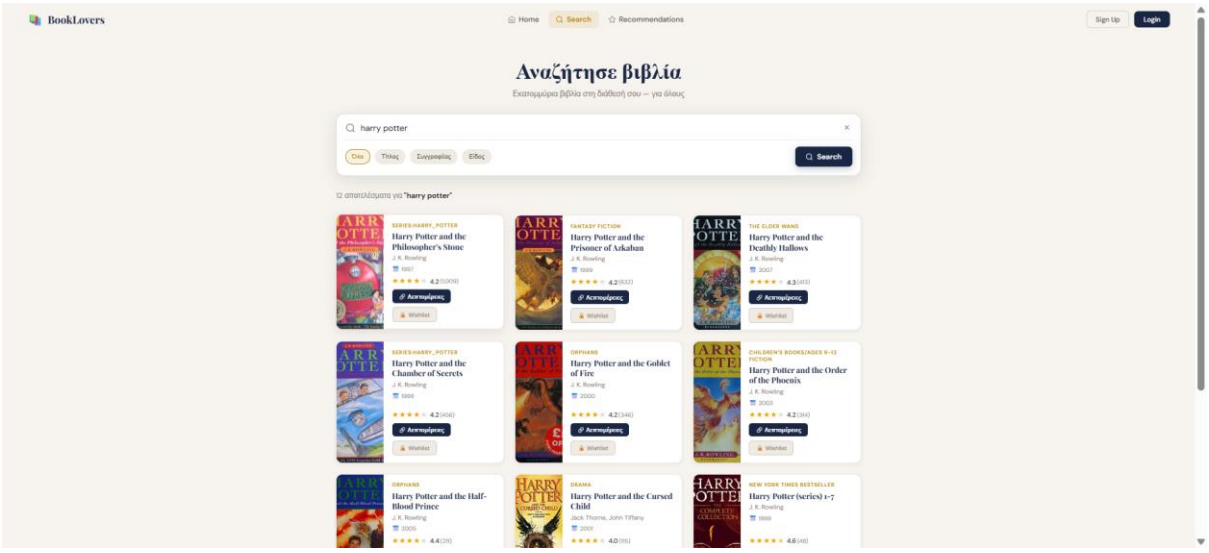
βιβλία: The Great Gatsby του F. Scott Fitzgerald με βαθμολογία 4.7, 1984 του George Orwell με βαθμολογία 4.9, Dune του Frank Herbert με βαθμολογία 4.8, Beloved της Toni Morrison με βαθμολογία 4.6, The Alchemist του Paulo Coelho με βαθμολογία 4.5 και Frankenstein της Mary Shelley με βαθμολογία 4.4. Στην επάνω δεξιά γωνία της ενότητας υπάρχει ο σύνδεσμος «Δες όλα →» ο οποίος οδηγεί τον χρήστη στη σελίδα αναζήτησης για να εξερευνήσει περισσότερους τίτλους.Αξίζει να σημειωθεί ότι το κουμπί «+ Wishlist» που εμφανίζεται σε κάθε κάρτα βιβλίου είναι διαθέσιμο οπτικά και στον μη εγγεγραμμένο χρήστη, ωστόσο εάν αυτός επιχειρήσει να το πατήσει χωρίς να είναι συνδεδεμένος, το σύστημα δεν καταχωρεί την ενέργεια και τον ανακατευθύνει στη σελίδα σύνδεσης.Στο δεύτερο τμήμα, στο κάτω μέρος της σελίδας, εμφανίζεται ένα σκουρόχρωμο banner με το μήνυμα «Θέλεις προσωπικές προτάσεις;», το οποίο προτρέπει τον επισκέπτη να δημιουργήσει λογαριασμό ή να συνδεθεί προκειμένου να αποκτήσει πρόσβαση στις εξατομικευμένες προτάσεις βιβλίων. Το banner αυτό συνοδεύεται από δύο κουμπιά: το κουμπί «Δημιούργησε λογαριασμό» που οδηγεί στη σελίδα εγγραφής και το κουμπί «Σύνδεση» που οδηγεί στη σελίδα σύνδεσης.



Εικόνα 7 – Σελίδα Αναζήτησης χρηστών

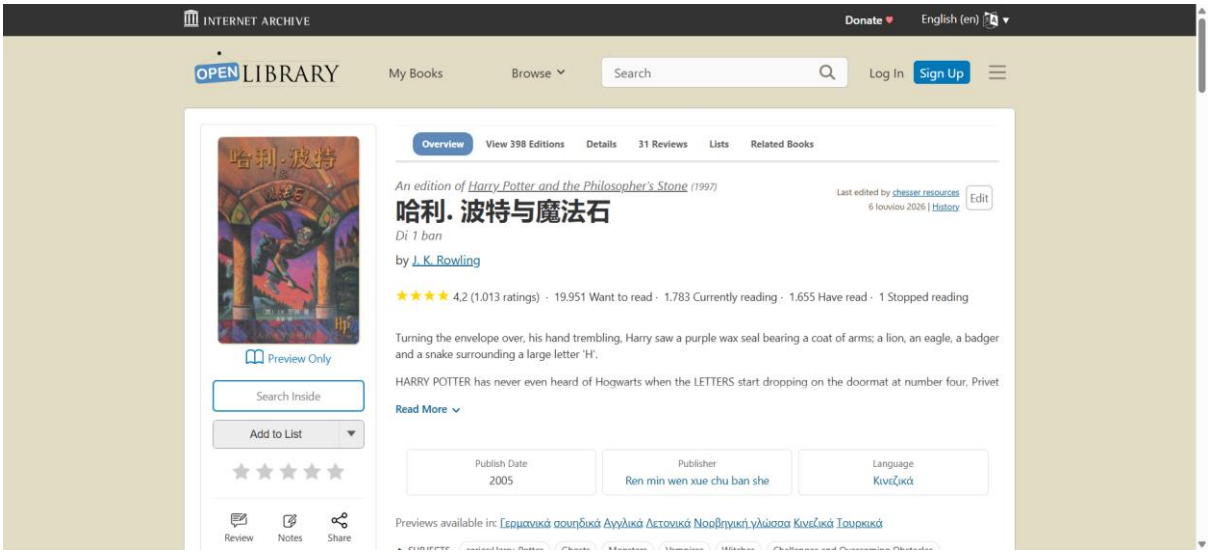
Μια από τις βασικές λειτουργίες και μοναδικές λειτουργίες που έχει στη διάθεσή του ο μη εγγεγραμμένος χρήστης είναι η αναζήτηση βιβλίων. Η αναζήτηση αυτή μπορεί να πραγματοποιηθεί με δύο τρόπους. Ο πρώτος τρόπος είναι μέσω του πεδίου αναζήτησης που βρίσκεται στο κεντρικό τμήμα της αρχικής σελίδας, όπως παρουσιάστηκε στην πρώτη εικόνα. Ο δεύτερος τρόπος είναι μέσω της αυτόνομης σελίδας αναζήτησης, στην οποία ο χρήστης μεταβαίνει πατώντας τον σύνδεσμο «Search» στη μπάρα πλοήγησης, και η οποία απεικονίζεται στην παραπάνω εικόνα. Όπως φαίνεται η σελίδα αναζήτησης φέρει τον τίτλο «Αναζήτησε βιβλία» και τον υπότιτλο «Εκατομμύρια βιβλία στη διάθεσή σου — για όλους», υποδηλώνοντας ότι η λειτουργία αυτή είναι ελεύθερα προσβάσιμη σε κάθε επισκέπτη. Στο κεντρικό τμήμα της σελίδας βρίσκεται το πεδίο αναζήτησης με το placeholder «Τίτλος, συγγραφέας, είδος...», συνοδευόμενο από τα τέσσερα κουμπιά φιλτραρίσματος: Όλα, Τίτλος, Συγγραφέας και Είδος. Ο χρήστης επιλέγει το επιθυμητό κριτήριο, πληκτρολογεί τον όρο

αναζήτησής του και πατά το κουμπί «Search». Επιπλέον, κάτω από το πεδίο αναζήτησης εμφανίζονται δημοφιλείς ετικέτες αναζήτησης όπως Science Fiction, Mystery, Romance, Fantasy, Thriller, Historical Fiction, Harry Potter και Dune, τις οποίες ο χρήστης μπορεί να πατήσει απευθείας για γρήγορη αναζήτηση χωρίς να χρειαστεί να πληκτρολογήσει.
Μετά την εκτέλεση της αναζήτησης, εμφανίζονται τα αποτελέσματα με τη μορφή καρτών βιβλίων, καθεμία από τις οποίες περιλαμβάνει το εξώφυλλο, τον τίτλο, τον συγγραφέα, τη βαθμολογία και το λογοτεχνικό είδος. Ο μη εγγεγραμμένος χρήστης έχει τη δυνατότητα να περιηγηθεί στα αποτελέσματα και να ανοίξει τη σελίδα κάθε βιβλίου για να δει περισσότερες πληροφορίες. Ωστόσο, οι δυνατότητές του σταματούν εκεί, καθώς λειτουργίες όπως η προσθήκη βιβλίου στη λίστα επιθυμιών και η λήψη εξατομικευμένων προτάσεων απαιτούν την ύπαρξη ενεργού λογαριασμού στην πλατφόρμα.Μια αναζήτηση ενός τίτλου βιβλίου και πιθανά αποτελέσματα σαν μη εγγεγραμμένος χρήστης:



Εικόνα 8 Σελίδα αποτελεσμάτων αναζήτησης βιβλίων

Και η δυνατότητα περιήγησης στο εξίσου δυναμικό site για την αναζήτηση βιβλίου και ανάγνωσης του.

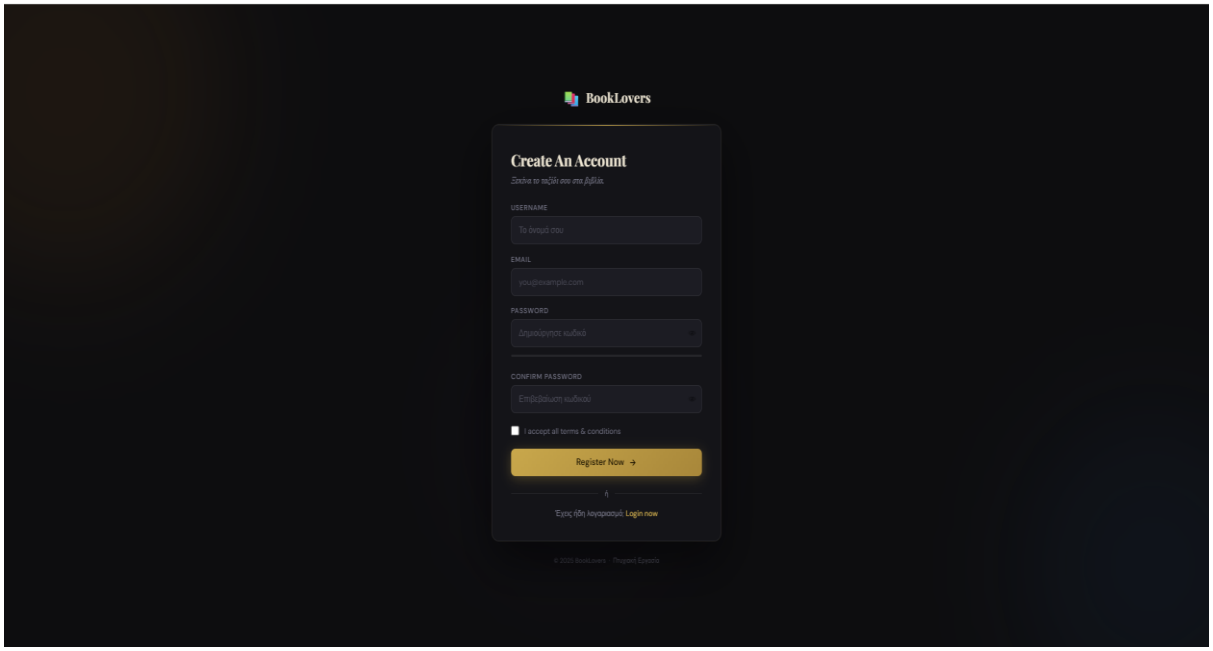


Εικόνα 9 Open Library ιστοσελίδα για ανάγνωση του βιβλίου

Η δυνατότητα για wishlist δεν μπορεί να πραγματοποιηθεί καθώς ζητείται από τον χρήστη να συνδεθεί η να φτιάξει λογαριασμό.

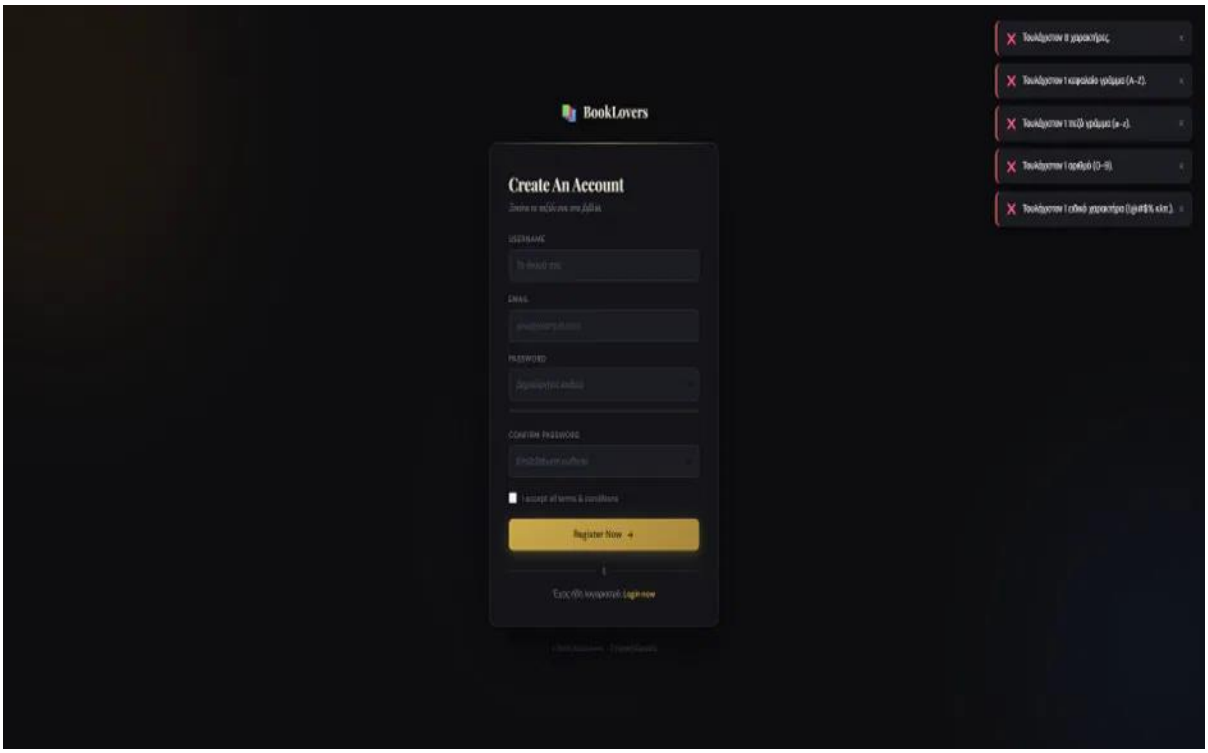
4.3 Εγχειρίδιο εγγεγραμμένου χρήστη

Στο παρόν υποκεφάλαιο παρουσιάζεται το εγχειρίδιο χρήσης της εφαρμογής από την πλευρά του εγγεγραμμένου χρήστη, δηλαδή του χρήστη που διαθέτει ενεργό λογαριασμό στην πλατφόρμα και έχει πραγματοποιήσει επιτυχή σύνδεση η του εν δυνάμει εγγρεγραμμένου χρήστη αυτού δηλαδή που θα δημιουργήσει λογαριασμό. Ο εγγεγραμμένος χρήστης αποτελεί τον κύριο αποδέκτη των δυνατοτήτων που προσφέρει η εφαρμογή BookLovers, καθώς μόνο μέσω του λογαριασμού του μπορεί να αξιοποιήσει πλήρως το σύστημα εξατομικευμένων προτάσεων.
Ο εγγεγραμμένος χρήστης μοιράζεται ορισμένες κοινές λειτουργίες με τον μη εγγεγραμμένο επισκέπτη. Συγκεκριμένα, η αρχική σελίδα της εφαρμογής, η σελίδα αναζήτησης βιβλίων και τα αποτελέσματα αναζήτησης είναι ταυτόσημα και για τους δύο τύπους χρηστών ως προς τη δομή και την εμφάνισή τους. Επιπλέον, και οι δύο τύποι χρηστών έχουν πρόσβαση στις δημοφιλείς ετικέτες αναζήτησης και στην ενότητα «Popular Books» της αρχικής σελίδας.
Ωστόσο, υπάρχουν σημαντικές διαφορές που διαχωρίζουν την εμπειρία του εγγεγραμμένου χρήστη από αυτή του μη εγγεγραμμένου. Αρχικά, η μπάρα πλοήγησης προσαρμόζεται αναλόγως, εμφανίζοντας το όνομα του συνδεδεμένου χρήστη και αντικαθιστώντας τα κουμπιά Sign Up και Login με το κουμπί αποσύνδεσης. Επιπλέον, ο σύνδεσμος Recommendations καθίσταται πλέον πλήρως λειτουργικός, παρέχοντας πρόσβαση στη σελίδα εξατομικευμένων προτάσεων. Το κουμπί «+ Wishlist» που εμφανίζεται σε κάθε κάρτα βιβλίου είναι πλέον ενεργό, επιτρέποντας στον χρήστη να αποθηκεύει βιβλία που τον ενδιαφέρουν. Τέλος, κάθε ενέργεια που πραγματοποιεί ο εγγεγραμμένος χρήστης, όπως αναζητήσεις, κλικ σε βιβλία και προσθήκες στο wishlist, καταγράφεται και συνδέεται με τον λογαριασμό του, διαμορφώνοντας σταδιακά ένα προφίλ προτιμήσεων που αξιοποιείται από τον αλγόριθμο προτάσεων.
Το πρώτο βήμα που καλείται να ακολουθήσει ο χρήστης προκειμένου να αποκτήσει πρόσβαση στις εξατομικευμένες δυνατότητες της εφαρμογής είναι η δημιουργία λογαριασμού. Η σελίδα εγγραφής, η οποία απεικονίζεται στην παρακάτω εικόνα , είναι προσβάσιμη είτε μέσω του κουμπιού «Sign Up» που βρίσκεται στη μπάρα πλοήγησης είτε μέσω του κουμπιού «Δημιούργησε λογαριασμό» που εμφανίζεται στο banner της αρχικής σελίδας.Όπως φαίνεται, η σελίδα εγγραφής διαθέτει σκούρο φόντο και εμφανίζει στο κέντρο της οθόνης μια φόρμα με τίτλο «Create An Account» και υπότιτλο «Ξεκίνα το ταξίδι σου στα βιβλία». Η φόρμα αποτελείται από τέσσερα πεδία εισαγωγής δεδομένων. Το πρώτο πεδίο αφορά το όνομα χρήστη (Username) και φέρει το placeholder «Το όνομά σου». Το δεύτερο πεδίο αφορά τη διεύθυνση ηλεκτρονικού ταχυδρομείου (Email) με placeholder «you@example.com». Το τρίτο πεδίο αφορά τον κωδικό πρόσβασης (Password) με placeholder «Δημιούργησε κωδικό», ενώ το τέταρτο πεδίο αφορά την επιβεβαίωση του κωδικού (Confirm Password) με placeholder «Επιβεβαίωση κωδικού».
Κάτω από τα πεδία εισαγωγής εμφανίζεται ένα πλαίσιο επιλογής (checkbox) με το μήνυμα «I accept all terms & conditions», το οποίο ο χρήστης οφείλει να επιλέξει πριν προχωρήσει στην εγγραφή. Στη συνέχεια, ο χρήστης πατά το κουμπί «Register Now» για να ολοκληρώσει τη διαδικασία δημιουργίας λογαριασμού. Εάν ο χρήστης διαθέτει ήδη λογαριασμό, μπορεί να μεταβεί στη σελίδα σύνδεσης πατώντας τον σύνδεσμο «Login now» που εμφανίζεται στο κάτω μέρος της φόρμας.



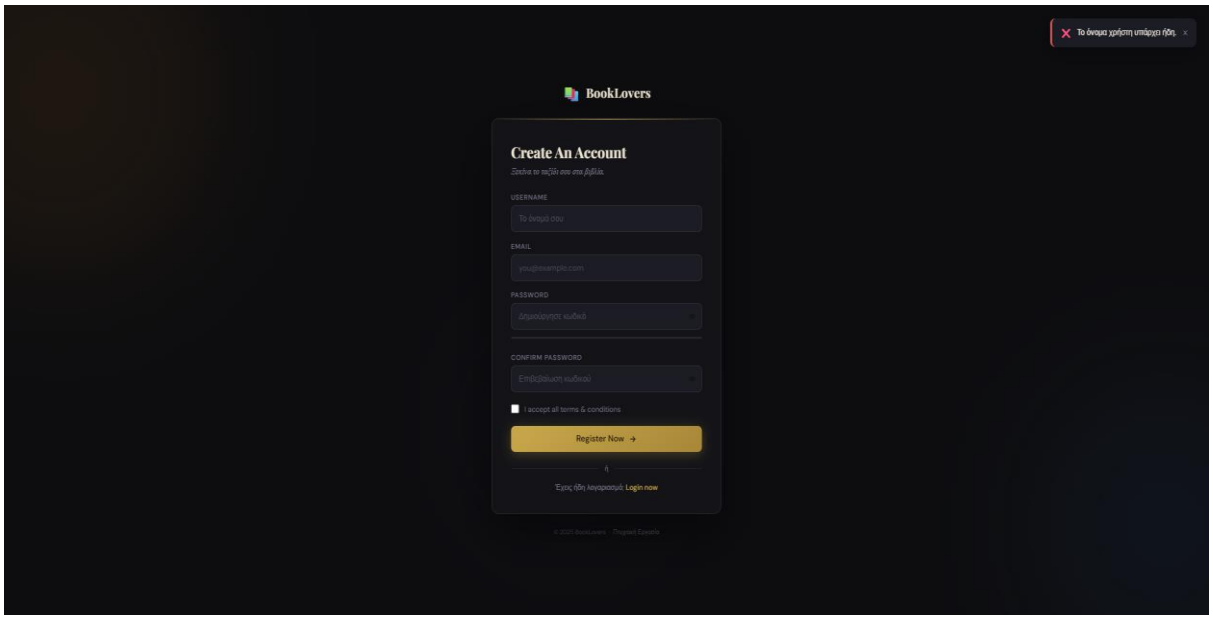
Εικόνα 10 Φόρμα δημιουργίας νέου χρήστη

Εάν ο χρήστης εισάγει κωδικό πρόσβασης που δεν πληροί τις απαιτήσεις πολιτικής ασφαλείας της εφαρμογής και πατήσει το κουμπί «Register Now», το σύστημα εμφανίζει αυτόματα τα αντίστοιχα μηνύματα σφάλματος, όπως απεικονίζεται στην παρακάτω εικόνα. Τα μηνύματα αυτά εμφανίζονται στην επάνω δεξιά γωνία της οθόνης με κόκκινο φόντο και το σύμβολο «Χ», ώστε να είναι άμεσα ορατά από τον χρήστη.
Συγκεκριμένα, τα μηνύματα σφάλματος που εμφανίζονται αφορούν τους πέντε κανόνες πολιτικής κωδικού πρόσβασης που έχουν υλοποιηθεί στην εφαρμογή. Το πρώτο μήνυμα ενημερώνει τον χρήστη ότι ο κωδικός πρέπει να αποτελείται από τουλάχιστον 8 χαρακτήρες. Το δεύτερο μήνυμα αναφέρει ότι απαιτείται τουλάχιστον ένα κεφαλαίο γράμμα (A-Z). Το τρίτο μήνυμα αναφέρει ότι απαιτείται τουλάχιστον ένα πεζό γράμμα (α-ζ). Το τέταρτο μήνυμα αναφέρει ότι απαιτείται τουλάχιστον ένας αριθμός (0-9). Το πέμπτο και τελευταίο μήνυμα αναφέρει ότι απαιτείται τουλάχιστον ένας ειδικός χαρακτήρας όπως !@#$% κλπ. Ο χρήστης οφείλει να διορθώσει τον κωδικό του ώστε να πληροί όλες τις παραπάνω απαιτήσεις πριν μπορέσει να ολοκληρώσει επιτυχώς τη διαδικασία εγγραφής. Στην περίπτωση που ο χρήστης εισάγει όνομα χρήστη ή διεύθυνση ηλεκτρονικού ταχυδρομείου που χρησιμοποιείται ήδη από άλλο εγγεγραμμένο χρήστη στην εφαρμογή, το σύστημα εμφανίζει αντίστοιχο μήνυμα σφάλματος ενημερώνοντάς τον ότι το όνομα χρήστη ή το email υπάρχει ήδη.



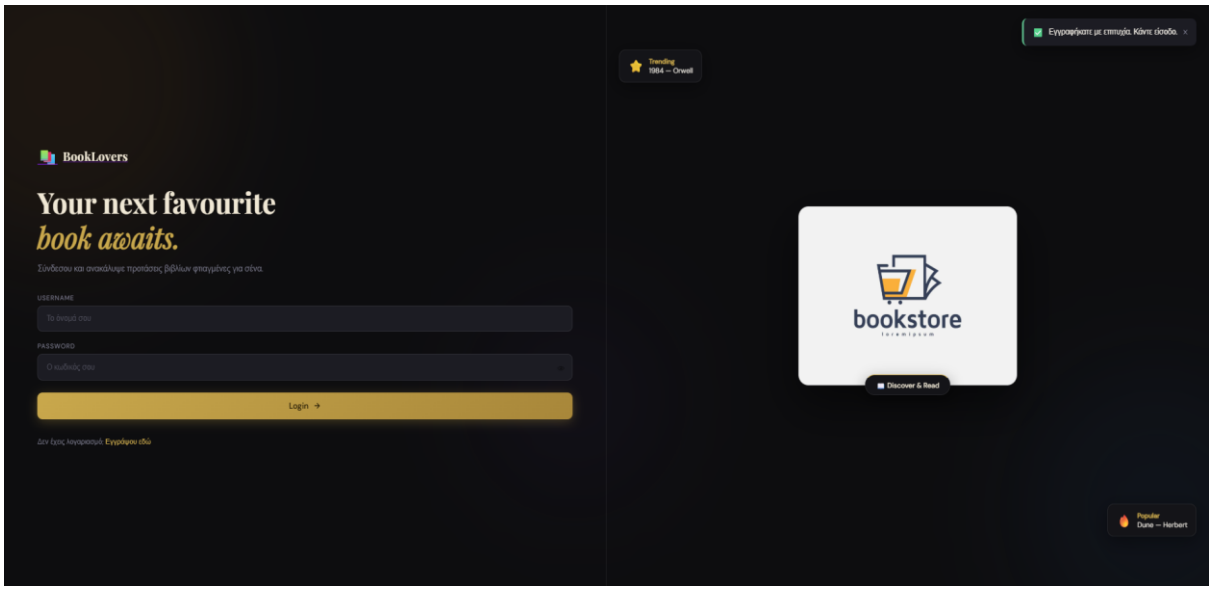
Εικόνα 11 Ελλιπή στοιχεία κατά την δημιουργία νέου χρήστη

Στην περίπτωση αυτή ο χρήστης καλείται να επιλέξει διαφορετικό όνομα χρήστη ή να χρησιμοποιήσει διαφορετική διεύθυνση ηλεκτρονικού ταχυδρομείου προκειμένου να ολοκληρώσει επιτυχώς τη διαδικασία εγγραφής. Το μήνυμα αυτό αποτρέπει τη δημιουργία διπλότυπων λογαριασμών στη βάση δεδομένων, εξασφαλίζοντας τη μοναδικότητα κάθε χρήστη στην πλατφόρμα.



Εικόνα 12 Ο χρήστης υπάρχει ήδη

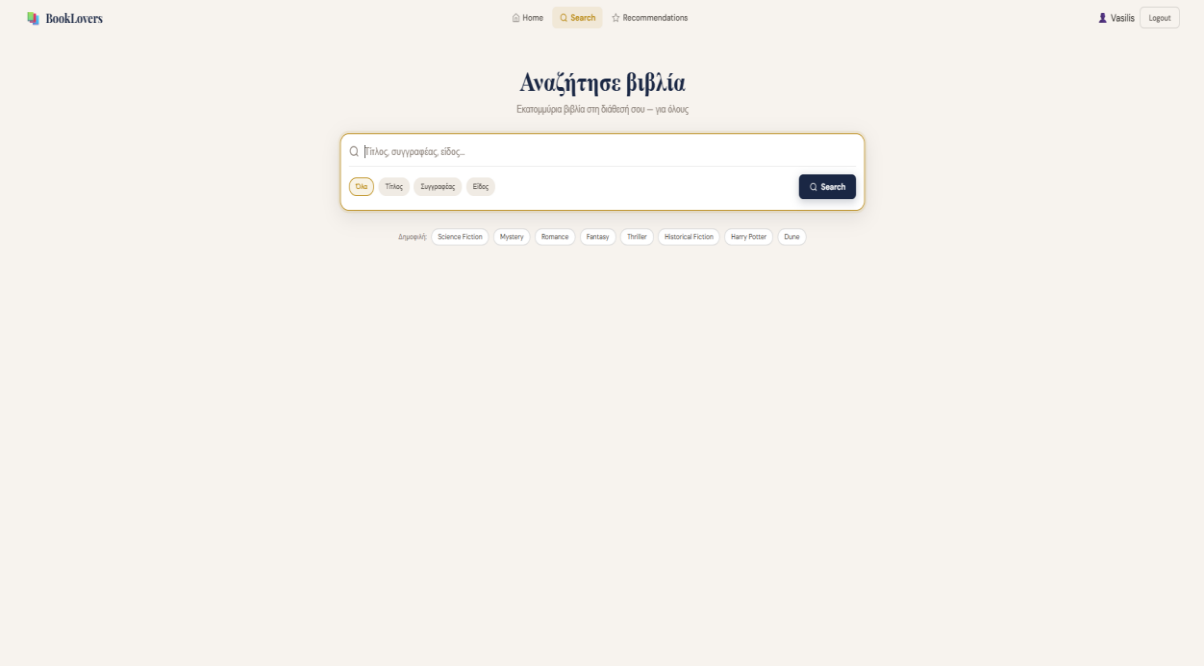
Μετά την επιτυχή εγγραφή, το σύστημα εμφανίζει στην επάνω δεξιά γωνία της οθόνης ένα πράσινο μήνυμα επιβεβαίωσης με το κείμενο «Εγγραφήκατε με επιτυχία. Κάντε είσοδο.» και ανακατευθύνει αυτόματα τον χρήστη στη σελίδα σύνδεσης, η οποία απεικονίζεται στην εικόνα μας.
Η σελίδα σύνδεσης, η οποία είναι η ακόλουθη και δεν προυποθέτει δημιουργία χρήστη αν έχει δημιουργηθεί ήδη λογαριασμός, διαθέτει επίσης σκούρο φόντο και είναι χωρισμένη σε δύο τμήματα. Στο αριστερό τμήμα εμφανίζεται η φόρμα σύνδεσης με τίτλο «Your next favourite book awaits.» και υπότιτλο «Συνδέσου και ανακάλυψε προτάσεις βιβλίων φτιαγμένες για σένα.». Η φόρμα αποτελείται από δύο πεδία εισαγωγής: το πεδίο Username με placeholder «Το όνομά σου» και το πεδίο Password με placeholder «Ο κωδικός σου». Κάτω από τα πεδία βρίσκεται το κουμπί «Login →» μέσω του οποίου ο χρήστης ολοκληρώνει τη σύνδεσή του. Εάν ο χρήστης δεν διαθέτει ακόμα λογαριασμό, μπορεί να μεταβεί στη σελίδα εγγραφής πατώντας τον σύνδεσμο «Εγγράψου εδώ» που εμφανίζεται στο κάτω μέρος της φόρμας.
Στο δεξί τμήμα της σελίδας εμφανίζεται μια διακοσμητική εικόνα bookstore, συνοδευόμενη από δύο ενδεικτικές κάρτες βιβλίων στις γωνίες της οθόνης. Συγκεκριμένα, στην επάνω αριστερή γωνία εμφανίζεται μια κάρτα με την ένδειξη «Trending» και το βιβλίο «1984 — Orwell», ενώ στην κάτω δεξιά γωνία εμφανίζεται μια κάρτα με την ένδειξη «Popular» και το βιβλίο «Dune — Herbert». Τα στοιχεία αυτά προσδίδουν στη σελίδα σύνδεσης έναν ελκυστικό και θεματικά συνεπή χαρακτήρα που εναρμονίζεται με τον σκοπό της εφαρμογής.



Εικόνα 13 Σύνδεση χρήστη

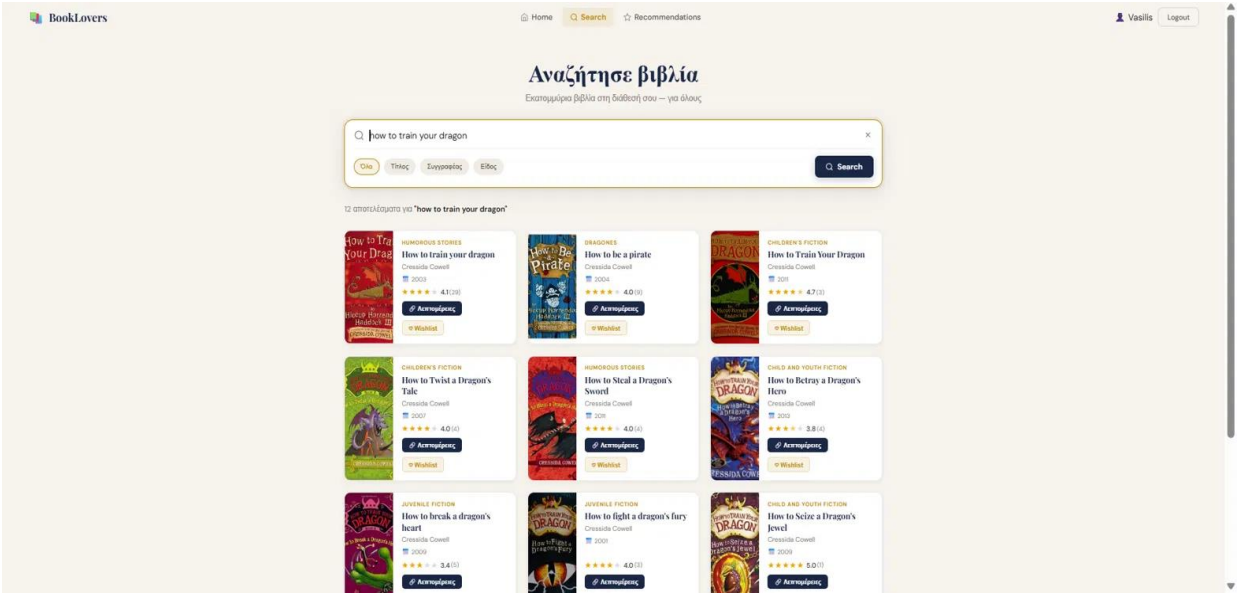
Μετά την επιτυχή σύνδεση, το σύστημα ανακατευθύνει αυτόματα τον χρήστη στην αρχική σελίδα της εφαρμογής. Η αρχική σελίδα του εγγεγραμμένου χρήστη είναι οπτικά ταυτόσημη με αυτή του μη εγγεγραμμένου επισκέπτη, με μια σημαντική διαφορά που εντοπίζεται στη μπάρα πλοήγησης. Συγκεκριμένα, τα κουμπιά Sign Up και Login που εμφανίζονταν στον ανώνυμο επισκέπτη έχουν αντικατασταθεί από το όνομα του συνδεδεμένου χρήστη και το κουμπί «Logout», επιβεβαιώνοντας την επιτυχή σύνδεση στο σύστημα.
Το γεγονός αυτό είναι εμφανές και στη σελίδα αναζήτησης, όπως απεικονίζεται στην εικόνα. Στην επάνω δεξιά γωνία της μπάρας πλοήγησης εμφανίζεται το εικονίδιο χρήστη

συνοδευόμενο από το όνομα του συνδεδεμένου χρήστη, στη συγκεκριμένη περίπτωση «Vasilis», καθώς και το κουμπί «Logout» για την αποσύνδεση από την εφαρμογή. Η σελίδα αναζήτησης παραμένει οπτικά ίδια με αυτή που βλέπει ο μη εγγεγραμμένος χρήστης, με τη διαφορά ότι πλέον οι ενέργειες του χρήστη καταγράφονται και συνδέονται με τον λογαριασμό του, συμβάλλοντας στη διαμόρφωση του προφίλ προτιμήσεών του.



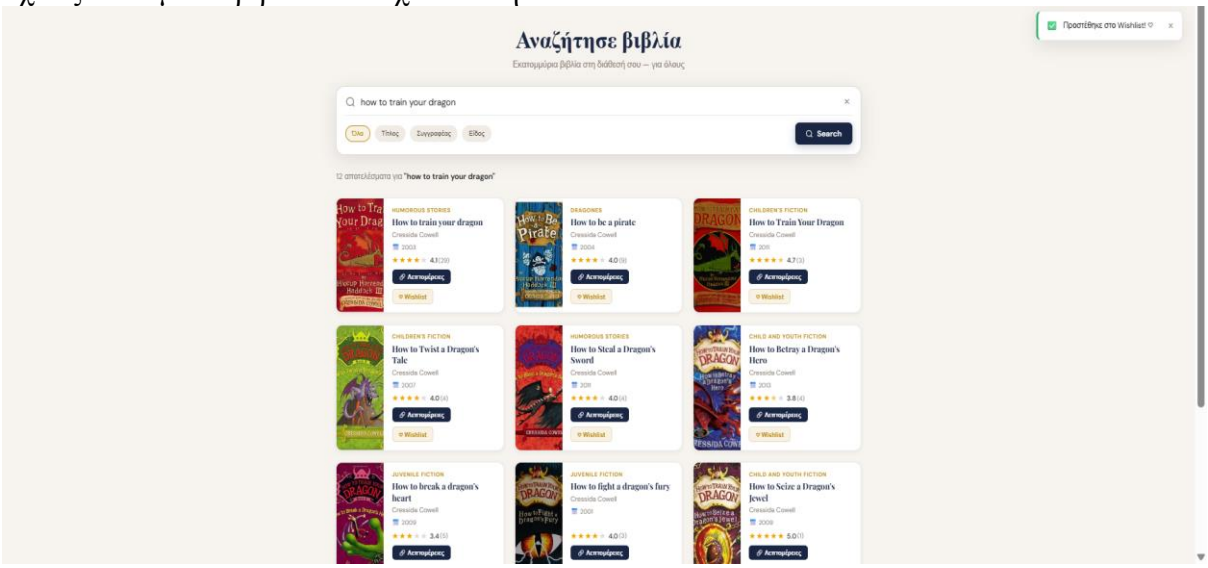
Εικόνα 14 Επιτυχής Σύνδεση και επιστροφή στην σελίδα αναζήτησης

Όπως φαίνεται στην ακόλουθη εικόνα, ο εγγεγραμμένος χρήστης έχει πραγματοποιήσει αναζήτηση με τον όρο «how to train your dragon» επιλέγοντας το κριτήριο «Όλα». Το σύστημα εμφάνισε 12 και περισσότερα αποτελέσματα, τα οποία παρουσιάζονται σε μορφή πλέγματος τριών στηλών. Κάθε κάρτα βιβλίου περιλαμβάνει το εξώφυλλο, το λογοτεχνικό είδος, τον τίτλο, τον συγγραφέα, το έτος έκδοσης, τη βαθμολογία με αστέρια και τον αριθμό αξιολογήσεων σε παρένθεση.
Το πρώτο είναι το κουμπί «Λεπτομέρειες» το οποίο οδηγεί τον χρήστη στη σελίδα του βιβλίου στο Open Library για να διαβάσει περισσότερες πληροφορίες, ενώ παράλληλα καταγράφει το γεγονός ως κλικ (clicked) με βάρος 2.0 στο προφίλ προτιμήσεων του χρήστη. Το δεύτερο είναι το κουμπί «Wishlist» το οποίο επιτρέπει στον χρήστη να αποθηκεύσει το βιβλίο στη λίστα επιθυμιών του, καταγράφοντας το γεγονός ως wishlist με βάρος 3.0. Και οι δύο αυτές ενέργειες συμβάλλουν στη διαμόρφωση του προφίλ προτιμήσεων του χρήστη, το οποίο αξιοποιείται από τον αλγόριθμο για την παραγωγή εξατομικευμένων προτάσεων.



Εικόνα 15 Εμφάνιση ονόματος χρήστη στα αποτελέσματα αναζήτησης βιβλίων

Όταν ο εγγεγραμμένος χρήστης πατήσει το κουμπί «Wishlist» σε κάποιο βιβλίο που τον ενδιαφέρει, το σύστημα εμφανίζει αμέσως ένα πράσινο μήνυμα επιβεβαίωσης στην επάνω δεξιά γωνία της οθόνης με το κείμενο «Προστέθηκε στο Wishlist!», όπως απεικονίζεται στην εικόνα. Το μήνυμα αυτό εξαφανίζεται αυτόματα μετά από λίγα δευτερόλεπτα, επιτρέποντας στον χρήστη να συνεχίσει την περιήγησή του στα αποτελέσματα αναζήτησης χωρίς διακοπή. Η ενέργεια αυτή καταγράφεται αυτόματα στο σύστημα ως γεγονός τύπου wishlist με βάρος 3.0, το οποίο είναι το υψηλότερο βάρος που αποδίδεται σε οποιαδήποτε ενέργεια του χρήστη εντός της εφαρμογής. Το υψηλό αυτό βάρος αντικατοπτρίζει το ισχυρό ενδιαφέρον που εκφράζει ο χρήστης για το συγκεκριμένο βιβλίο, καθώς η προσθήκη στο wishlist αποτελεί μια συνειδητή και σκόπιμη ενέργεια που υποδηλώνει σαφή πρόθεση ανάγνωσης. Το γεγονός αυτό αξιοποιείται άμεσα από τον αλγόριθμο προτάσεων, ο οποίος το λαμβάνει υπόψιν κατά την επόμενη επίσκεψη του χρήστη στη σελίδα Recommendations, παράγοντας προτάσεις που σχετίζονται με τα βιβλία που έχει αποθηκεύσει.



Εικόνα 16 Προσθήκη βιβλίου στο wishlist

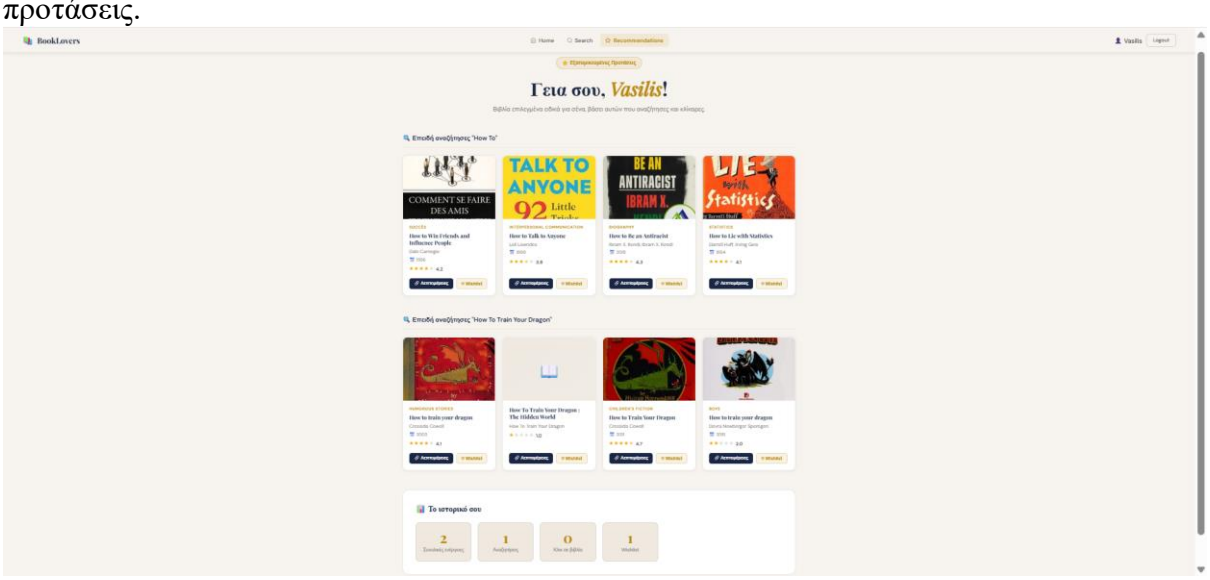
Η τελευταία και σημαντικότερη λειτουργία που έχει στη διάθεσή του ο εγγεγραμμένος χρήστης είναι η σελίδα Recommendations, η οποία είναι προσβάσιμη μέσω του αντίστοιχου συνδέσμου στη μπάρα πλοήγησης. Η σελίδα αυτή αποτελεί τον πυρήνα της εφαρμογής BookLovers και παρουσιάζει εξατομικευμένες προτάσεις βιβλίων που έχουν παραχθεί αποκλειστικά για τον συγκεκριμένο χρήστη.

Στο επάνω μέρος της σελίδας εμφανίζεται ένα προσωποποιημένο χαιρετισμός με το μήνυμα «Γεια σου, Vasilis!», συνοδευόμενο από τον υπότιτλο «Βιβλία επιλεγμένα ειδικά για σένα, βάσει αυτών που αναζήτησες και κλίκαρες». Το μήνυμα αυτό αντικατοπτρίζει τον εξατομικευμένο χαρακτήρα της σελίδας, καθώς το όνομα του χρήστη αντλείται δυναμικά από τον λογαριασμό του.

Στη συνέχεια εμφανίζονται οι ομάδες προτάσεων, καθεμία από τις οποίες συνοδεύεται από αιτιολόγηση που εξηγεί στον χρήστη γιατί του προτείνονται τα συγκεκριμένα βιβλία. Στη συγκεκριμένη περίπτωση εμφανίζονται δύο ομάδες προτάσεων. Η πρώτη φέρει την ένδειξη «Επειδή αναζήτησες 'How To'» και περιλαμβάνει τέσσερα βιβλία σχετικά με τον όρο αυτό, όπως το «How to Win Friends and Influence People» του Dale Carnegie και το «How to Talk to Anyone» της Leil Lowndes. Η δεύτερη ομάδα φέρει την ένδειξη «Επειδή αναζήτησες 'How To Train Your Dragon'» και περιλαμβάνει τέσσερα βιβλία της σειράς αυτής.

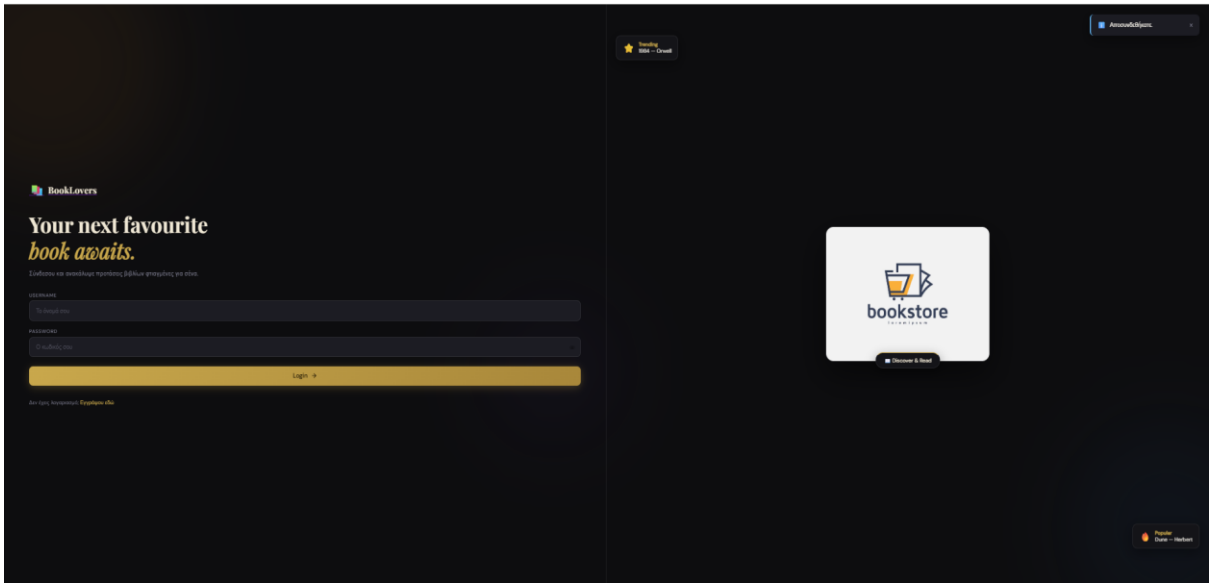
Αξίζει να τονιστεί ότι οι προτάσεις αυτές δεν είναι στατικές αλλά ανανεώνονται δυναμικά κάθε φορά που ο χρήστης επισκέπτεται τη σελίδα. Συγκεκριμένα, κάθε νέα αναζήτηση, κάθε κλικ σε βιβλίο και κάθε προσθήκη στο wishlist που πραγματοποιεί ο χρήστης καταγράφεται αυτόματα και λαμβάνεται υπόψιν από τον αλγόριθμο κατά την επόμενη επίσκεψη στη σελίδα προτάσεων, με αποτέλεσμα οι προτάσεις να εξελίσσονται και να βελτιώνονται συνεχώς όσο περισσότερο χρησιμοποιεί ο χρήστης την εφαρμογή.

Στο κάτω μέρος της σελίδας εμφανίζεται η ενότητα «Το ιστορικό σου», η οποία παρουσιάζει συνοπτικά στατιστικά στοιχεία για τη δραστηριότητα του χρήστη εντός της εφαρμογής. Συγκεκριμένα εμφανίζονται τέσσερις κάρτες με τον συνολικό αριθμό ενεργειών, τον αριθμό αναζητήσεων, τον αριθμό κλικ σε βιβλία και τον αριθμό προσθηκών στο wishlist. Στη συγκεκριμένη περίπτωση ο χρήστης έχει πραγματοποιήσει συνολικά 2 ενέργειες, εκ των οποίων 1 αναζήτηση, 0 κλικ σε βιβλία και 1 προσθήκη στο wishlist. Τα στατιστικά αυτά βοηθούν τον χρήστη να κατανοήσει πόσο εξατομικευμένες είναι οι προτάσεις που λαμβάνει και τον ενθαρρύνουν να αλληλεπιδρά περισσότερο με την εφαρμογή για ακόμα πιο ακριβείς προτάσεις.



Εικόνα 17 Σελίδα προτάσεων και προτιμήσεων βιβλίων

Τέλος, όταν ο χρήστης επιθυμεί να αποσυνδεθεί από την εφαρμογή, πατά το κουμπί «Logout» που βρίσκεται στην επάνω δεξιά γωνία της μπάρας πλοήγησης. Το σύστημα εμφανίζει αμέσως ένα μήνυμα επιβεβαίωσης στην επάνω δεξιά γωνία της οθόνης με το κείμενο «Αποσυνδεθήκατε.» και ανακατευθύνει αυτόματα τον χρήστη στη σελίδα σύνδεσης, όπως απεικονίζεται στην παραπάνω εικόνα.
Ένα ιδιαίτερα σημαντικό χαρακτηριστικό της εφαρμογής BookLovers είναι ότι όλες οι ενέργειες που έχει πραγματοποιήσει ο χρήστης κατά τη διάρκεια της συνεδρίας του διατηρούνται πλήρως στη βάση δεδομένων και δεν χάνονται με την αποσύνδεση. Συγκεκριμένα, οι αναζητήσεις, τα κλικ σε βιβλία και οι προσθήκες στο wishlist που έχουν καταγραφεί παραμένουν αποθηκευμένα και συνδεδεμένα με τον λογαριασμό του χρήστη. Επομένως, την επόμενη φορά που ο χρήστης θα συνδεθεί εκ νέου στην εφαρμογή, το σύστημα θα αναγνωρίσει αυτόματα το ιστορικό των ενεργειών του και θα παράγει προτάσεις που λαμβάνουν υπόψιν το σύνολο της δραστηριότητάς του, ακόμα και από προηγούμενες συνεδρίες. Με αυτόν τον τρόπο το προφίλ προτιμήσεων του χρήστη εξελίσσεται και εμπλουτίζεται συνεχώς με κάθε νέα επίσκεψη στην εφαρμογή, καθιστώντας τις προτάσεις ολοένα και πιο ακριβείς και στοχευμένες.



Εικόνα 18 Αποσύνδεση από την εφαρμογή

Κεφάλαιο 5ο

5 Συμπεράσματα

Από το σύνολο των δεδομένων που παρατέθηκαν σε όλη την έκταση της παρούσας επιστημονικής εργασίας, γίνεται κατανοητό ότι η διαδικτυακή εφαρμογή BookLovers αποτελεί μια ολοκληρωμένη πλατφόρμα παροχής εξατομικευμένων προτάσεων βιβλίων βασισμένη σε τεχνικές μηχανικής μάθησης. Η ιδέα για την υλοποίησή της προέκυψε από την ανάγκη αντιμετώπισης ενός διαδεδομένου προβλήματος που αντιμετωπίζουν οι σύγχρονοι αναγνώστες, το οποίο δεν είναι άλλο από τη δυσκολία επιλογής του κατάλληλου βιβλίου μέσα από το τεράστιο πλήθος των διαθέσιμων τίτλων, σε μια εποχή που η ανάγνωση βιβλίων υποχωρεί έναντι των ψηφιακών μέσων ψυχαγωγίας. Αν και υφίστανται παρόμοιες εφαρμογές, όπως το StoryGraph, το WhichBook και το WhatShouldIReadNext, οι δυνατότητες και οι λειτουργίες που παρέχουν δεν επαρκούν για να προσφέρουν πλήρως εξατομικευμένη εμπειρία, καθώς παρουσιάζουν περιορισμούς που αφορούν τη γλώσσα, την έλλειψη προηγμένων αλγορίθμων μηχανικής μάθησης ή την απουσία δυναμικής καταγραφής της συμπεριφοράς του χρήστη. Για τον λόγο αυτό κρίθηκε αναγκαία η δημιουργία της εφαρμογής BookLovers, η οποία αντιμετωπίζει τις αδυναμίες αυτές αξιοποιώντας ένα υβριδικό σύστημα προτάσεων που συνδυάζει content-based filtering μέσω TF-IDF.Η εφαρμογή παρέχει πρόσβαση τόσο σε μη εγγεγραμμένους χρήστες, οι οποίοι μπορούν να αναζητούν βιβλία μέσω του Open Library API, όσο και σε εγγεγραμμένους χρήστες, οι οποίοι απολαμβάνουν πλήρως εξατομικευμένες προτάσεις βάσει του ιστορικού των ενεργειών τους. Εν κατακλείδι, το σύνολο των στοιχείων που έχουν παρατεθεί καθιστά τη BookLovers μια ολοκληρωμένη και εύχρηστη διαδικτυακή εφαρμογή που συνδυάζει με επιτυχία την τεχνολογία της μηχανικής μάθησης με την απλότητα χρήσης, αποτελώντας ένα αξιόλογο εργαλείο για κάθε αναγνώστη που επιθυμεί να ανακαλύψει το επόμενο βιβλίο του.

6. Βιβλιογραφικές Πηγές

1.Open Library. Internet Archive. Διαθέσιμο στον διαδικτυακό τόπο: https://openlibrary.org/

2.Open Library. API Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://openlibrary.org/developers/api

3.Python Software Foundation. Python 3 Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://docs.python.org/3/

4.Pallets Projects. Flask Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://flask.palletsprojects.com/

5.Pallets Projects. Flask-SQLAlchemy Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://flask-sqlalchemy.palletsprojects.com/

6.Bauer M. Flask-Login Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://flask-login.readthedocs.io/

7.SQLite. About SQLite. Διαθέσιμο στον διαδικτυακό τόπο: https://www.sqlite.org/index.html

8.Scikit-learn. Machine Learning in Python. Διαθέσιμο στον διαδικτυακό τόπο: https://scikit-learn.org/

9.Scikit-learn. TfidfVectorizer Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

10.Scikit-learn. Cosine Similarity Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

11.Requests. Requests: HTTP for Humans. Διαθέσιμο στον διαδικτυακό τόπο: https://requests.readthedocs.io/

12.NumPy. NumPy Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://numpy.org/doc/

13.Werkzeug. Werkzeug Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://werkzeug.palletsprojects.com/

14.JetBrains. PyCharm Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://www.jetbrains.com/pycharm/

15.StoryGraph. The StoryGraph. Διαθέσιμο στον διαδικτυακό τόπο: https://www.thestorygraph.com/

16.WhichBook. WhichBook. Διαθέσιμο στον διαδικτυακό τόπο: https://www.whichbook.net/

17.WhatShouldIReadNext. What Should I Read Next. Διαθέσιμο στον διαδικτυακό τόπο: https://www.whatshouldireadnext.com/

18.Jinja. Jinja2 Template Engine Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://jinja.palletsprojects.com/

19.Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge. Cambridge University Press, 2008. Διαθέσιμο στον διαδικτυακό τόπο: https://nlp.stanford.edu/IR-book/

20.Aggarwal C.C. Recommender Systems: The Textbook. New York. Springer, 2016.

21.Burke R. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 2002. Διαθέσιμο στον διαδικτυακό τόπο: https://link.springer.com/article/10.1023/A:1021240730564

22.SQLAlchemy. SQLAlchemy Documentation. Διαθέσιμο στον διαδικτυακό τόπο: https://docs.sqlalchemy.org/