HY-359 Project 2022 E-libraries

Κουμάκης Εμμανουήλ
AM : 4281
Team id : 34

Εισαγωγή:

Το E-Library359 αποτελεί ένα διαδικτυακό πληροφοριακό σύστημα το οποίο σχεδιάστηκε και υλοποιήθηκε το χειμερινό εξάμηνο του 2022 στα πλαίσια του μαθήματος HY-359 - Διαδικτυακός Προγραμματισμός.

Σκοπός του πληροφοριακού συστήματος είναι να δίνει την δυνατότητα στους φοιτητές που σπουδάζουν στην Κρήτη να δανείζονται βιβλία από βιβλιοθήκες σε όλη την Κρήτη. Υποστηρίζει τρεις τύπους χρηστών :
1.Φοιτητές οι οποίοι μπορούν να δανείζονται και να γράφουν κριτικές σε βιβλία που έχουν δανειστεί στο παρελθόν.
2.Υπεύθυνους βιβλιοθήκης που διαχειρίζονται μια βιβλιοθήκη (Προσθήκη νέων βιβλίων κλπ).
3.Απλούς επισκέπτες που μπορούν να αναζητήσουν βιβλία και να δουν πληροφορίες για αυτά, όμως δεν μπορούν να τα δανειστούν.

Τέλος υπάρχει και ένας διαχειριστής του συστήματος ο οποίος έχει πρόσβαση από διαφορετική σελίδα η οποία είναι κρυφή στους χρήστες και εχει την δυνατότητα να βλέπει στατιστικά σχετικά με το σύστημα, να διαχειρίζεται τους χρήστες, να βγάζει ανακοινώσεις στην κύρια σελίδα κλπ.

Συνοπτικά το client-side της εφαρμογής υλοποιήθηκε μέσω javascript και ajax requests στον server και το server-side μέσω java servlets.Η επικοινωνία τους πραγματοποιήθηκε κυρίως στέλνοντας και λαμβάνοντας json αρχεία ο ένας στον άλλο. Αυτά τα json strings τα επεξεργαζόμαστε κατάλληλα και ανανεώνουμε το περιεχόμενο της σελίδα χωρίς να χρειάζεται να κάνουμε

reload ολόκληρο το page (ανανεώνουμε δηλαδή τον ήδη υπάρχον κώδικα html και δεν φορτώνουμε καινούριο). Λεπτομέρειες για την υλοποίηση κάθε μερίας αναφέρονται πιο αναλυτικά παρακάτω.

Γενικά:

Από την μεριά του client στέλνουμε και λαμβάνουμε πληροφορίες από τον server μέσω ajax XMLHttpRequests και απο την μεριά του server τα διαχειριζόμαστε μέσω java servlets.

Συνολικά τα html αρχεία που χρειάστηκε να δημιουργήσω είναι 8, τα javascript αρχεία είναι 10 και ο αριθμός των servlets ειναι 28. Όλα αυτά είναι δομημένα σε φακέλους ανάλογα με το σε ποια κατηγορία χρειάζονται και περιγράφεται πιο αναλυτικά η χρήση τους παρακάτω. Επίσης είχα δημιουργήσει κατά την διάρκεια της 3ης άσκησης ένα αρχείο filterinput.java για να φιλτράρω το Input του χρήστη προτού προσθέσω/αλλάξω κάτι στην βάση και άλλο ένα αρχείο CheckDB.java για να πραγματοποιούνται αναζητήσεις και έλεγχοι στην βάση. Αυτά τα 2 αρχεία χρησιμοποιούνται ακόμα και βρίσκονται στο φάκελο /mainClasses. Επιπρόσθετα σε αυτό το φάκελο δημιούργησα ακόμη ένα αρχειο competitor.java το οποίο χρειαζόμαστε για το giveaway bonus.

Από εξωτερικές βιβλιοθήκες javascript χρησιμοποίησα jquery σε κάποια σημεία jquery για να κάνω πιο εύκολη την ζωή μου και να γράφω πιο γρήγορα javascript κώδικα.
Όσον αφορά το ajax πέρα από τα XML Requests χρησιμοποιήθηκε και σε κάποια σημεία για να φορτώνεται ασύγχρονα το περιεχόμενο της σελίδας.
REST API δεν χρησιμοποιήθηκε καθόλου καθώς είχα θέμα με την εγκατάσταση του απο την 3η άσκηση.
Προσπαθησα επίσης να χρησιμοποιήσω javascript βιβλιοθήκες όπως pdfmake, jspdf και html-pdf για την δημιουργία του pdf σε client side (παίρνοντας δηλαδή κώδικα html και να τον κάνω export σε pdf για να είναι πιο όμορφα δομημένο από ότι θα ήταν αν δημιουργούσα το pdf στην java) όμως δεν έβγαλα άκρη με καμία από αυτές τις βιβλιοθήκες.

Όσον αφορά το styling της σελίδας δεν έχω κάνει κάτι πολύ ιδιαίτερο. Έφτιαξα ένα header και ένα footer που παραμένουν (σχεδόν) ίδια σε κάθε σελίδα της εφαρμογής απλά και μόνο για να δίνεται μια λιγο πιο ρεαλιστική αισθητική.

Πέρα από αυτά τα δύο έχω πειράξει λίγο το style των κουμπιών, των πινάκων, των γραμματοσειρών και των labels, καθώς έχω κάνει και λίγο responsive το site σε κάποια σημεία.
Όλα τα παραπάνω πραγματοποιούνται σε ένα αρχείο /css/style.css

Τα APIs που χρησιμοποίησα είναι τα εξής :

- Google charts για οπτικοποιήσεις (chart pies).
- OpenLayers για την προβολή σημείου (διεύθυνσης) στον χάρτη (από την άσκηση 2).
- TrueWay Matrix για τον υπολογισμό απόστασης ανάμεσα στη διεύθυνση του φοιτητή και στις βιβλιοθήκες που έχουν διαθέσιμο το βιβλίο που θέλει να δανειστεί.

1)Αλλαγές στο σχήμα της βάσης χρειάστηκε να κάνω στον πίνακα borrowing, όπου πρόσθεσα μια στήλη 'library_id' που θα κρατάμε το id της βιβλιοθήκης από την οποία ο χρήστης έχει δανειστεί το βιβλίο. Το ξέρω ότι θα μπορούσαμε να το βρίσκουμε κάθε φορά από τον πίνακα booksinlibraries μέσω του κοινού κλειδιού 'borrowing id', αλλά με αυτόν τον τρόπο οι διαδικασίες "Ενημέρωση κατάστασης δανεισμού" και "Προβολή ενεργών δανεισμών" γίνονται πολύ πιο απλές ως προς την αναζήτηση της βιβλιοθήκης.
2)Μια ακόμη αλλαγή που χρειάστηκε να κάνω για την υλοποίηση του bonus ερωτήματος με τα giveaways, ήταν στον πίνακα students όπου πρόσθεσα μια επιπλέον στήλη 'borrow_count' για να μετράμε πόσα βιβλία έχει δανειστεί συνολικά ο κάθε φοιτητής. (αυτός ο μετρητής αυξάνεται και όταν κάνει κριτική σε ένα βιβλίο ο χρήστης).

Ανάλυση κάθε κατηγορίας:

1. Λειτουργίες Administrator:

Τα servlets που χρειάζονται για αυτό το κομμάτι βρίσκονται στο φάκελο java/servlets/AdminActions, ενώ από την μεριά του client έχουμε το javascript αρχείο AdminAjax.js, καθώς χρησιμοποιούμε και κάποιες βοηθητικές συναρτήσεις από το αρχείο utils.js

Πως είναι υλοποιημένη η κάθε λειτουργικότητα :

- Αρχικά ο admin έχει πρόσβαση στην σελίδα από διαφορετικό (κρυφό) url. Έτσι όταν πληκτρολογεί στον Browser την διεύθυνση http://localhost:8080/hy359_project/adminActions , το servlet AdminPage.java στελνει μέσω μιας συνάρτησης doGet μια ολόκληρη σελίδα html την οποία διαβάζει από το αρχείο Admin.html.

- Για να κάνει login ο admin χρησιμοποιούμε απλους js ελέγχους. (Δεν ζητάμε τίποτα από τον server αφού δεν είναι αποθηκευμένα τα στοιχεία του κάπου στην βάση).

- Όσον αφορά τις λειτουργίες για τα στοιχεία των φοιτητών και των βιβλιοθηκάριων αντίστοιχα, από την μεριά του client στέλνουμε μέσω ajax get request στον server και οταν πάρουμε την απαντηση σε json δημιουργούμε το πινακάκι με τα στοιχεία του μέσω της συνάρτησης `createAdminTable()`. Προσθέτουμε επίσης ένα delete κουμπί σε κάθε στήλη που όταν πατηθεί στέλνεται ένα delete request για την διαγραφή του αντίστοιχου χρήστη.
  Από την μερία του server όσον αφορά το πρώτο κομμάτι, στα servlets getStudents.java και getStudents.java παίρνουμε τους χρήστες απο την βάση και τους επιστρέφουμε στον client σε μορφή json string.
  Για το κομμάτι της διαγραφής εντοπίζουμε τον χρήστη στην βάση μέσω του username που πήραμε από το request και τον διαγράφουμε.

- Για τις λειτουργίες με τα στατιστικά από την μεριά του client στέλνουμε απλά ένα (διαφορετικό για κάθε περίπτωση) get request στον server (μέσω ajax οπως παντα) και όταν λάβουμε την απάντηση σε json, χρησιμοποιούμε την συνάρτηση drawChart() για να φτιάξουμε το pie chart μέσω του google chart API.
  Από την μεριά του server, στα servlets BookPerLib.java , BookPerGenre.java και GetStudentStats.java έχουμε στο καθένα μια μέθοδο που υπολογίζει/μετράει τον αριθμό εμφάνισης του κάθε ζητούμενου (αριθμος φοιτητών, αριθμος βιβλίων ενός είδους κτλ).
  Έχουμε μια δεύτερη μέθοδο που παίρνει από την βάση τα στοιχεία που χρειαζόμαστε (π.χ όλα τα είδη των βιβλίων) και για κάθε τέτοιο στοιχείο (π.χ είδος) το ομαδοποιεί (τα μετατρέπει σε json objects) με τον αριθμό που έχει υπολογίσει. Τέλος δημιουργουμε ενα json array με όλα τα προηγούμενα json objects το οποίο τελικά επιστρέφουμε στον client σε μορφή json string.

2. Λειτουργίες Επισκέπτη:

Δεν δημιούργησα καινούρια αρχεία συγκεκριμένα για αυτή την κατηγορία καθώς τις λειτουργίες εδω τις χρειαζόμαστε και σε άλλα σημεία της εφαρμογής.
Ας πούμε ότι τα html αρχεία αυτής της κατηγορίας είναι το index.html (το main αρχείο το οποίο ανανεώνεται κάθε φορά) και το HomePage.html το οποίο βλέπει κάθε χρήστης που θα μπει στην σελίδα (προτού κάνει κάποιο register ή login).
Το servlet που χρειάζεται για αυτό το κομμάτι είναι το java/servlets/Books/BookActions

Πως είναι υλοποιημένη η κάθε λειτουργικότητα :

- Προβολή βιβλίων : Από την μεριά του client μέσω XML get request στέλνουμε στον server μια φόρμα με τα φίλτρα για τα βιβλία που θέλει να δει ο χρήστης (είδος βιβλίου, αριθμός σελίδων κλπ). Λαμβάνει πίσω από τον server ένα json string που περιέχει τα σχετικά βιβλία και δημιουργείται ένα πινακάκι για κάθε τέτοιο βιβλίο με τα στοιχεία του. Από τη μεριά του server γίνονται οι απαραίτητοι έλεγχοι ορθότητας (ο αριθμος σελίδων να είναι θετικός κλπ) και στην συνέχεια αναζητούνται τα βιβλία στην βάση όπου και επιστρέφονται σε μορφή json string.

- Το ζητούμενο με τις πληροφορίες για άλλα sites/links υλοποιείται με απλό html κώδικα και την χρήση css. Έχω ένα fixed footer το οποίο είναι ορατό σε κάθε σημείο του site και περιέχει πάνω τις ζητούμενες πληροφορίες.

3. Λειτουργίες Υπεύθυνου βιβλιοθήκης:

Τα servlets που χρειάζονται για αυτό το κομμάτι βρίσκονται στο φάκελο java/servlets/librarian, αλλά χρησιμοποιούνται και άλλα κοινόχρηστα servlets που βρίσκονται στους φακέλους books και utilities (logout, ελεγχος για διπλα username κλπ).Από την μεριά του client, πέρα από το αρχείο

librarianActions.js όλα τα υπόλοιπα είναι επίσης κοινα (με αυτά για τους students) .

Πως είναι υλοποιημένη η κάθε λειτουργικότητα :

- Register : Από την μεριά του client ελέγχουμε την ορθότητα των στοιχείων στο αρχείο registration_check.js (το οποίο το έχω από την 2η ασκηση). Για να ελέγξουμε αν υπάρχει ήδη το email ή το username στέλνουμε ajax get request στον server και σε αυτόν κοιτάμε αν υπαρχει καπου στην βαση αυτο το username ή email και στελνουμε πίσω αντίστοιχο μήνυμα.
  Όταν ολοκληρωθούν όλοι οι έλεγχοι στέλνουμε την φόρμα στον server μέσω ajax post request και στον server αποθηκεύουμε τα στοιχεία του στην βάση.

- Login : Πάλι στέλνουμε το username και password μεσω post request. Επίσης όταν είναι επιτυχές το login κάνουμε bind στο session 2 cookies με το όνομα και το είδος του χρήστη μέσω του HttpSession API για να κρατάμε τον χρήστη συνδεδεμένο.
- Η ανανέωση στοιχείων είναι πάνω κάτω ίδια διαδικασία με το register τόσο από την μερια του server αλλά και του client.

- Καταχώρηση νέου βιβλίου : Από την μεριά του client στέλνουμε στον server μια φόρμα με τα στοιχεία του βιβλίου σε μορφή μορφή json string, μέσω ajax post request.
  Από την μεριά του server κάνουμε τους απαραίτητους ελέγχους (το isbn να μην υπάρχει ήδη, έτος κυκλοφορίας >1200 κλπ) και αν είναι όλα ορθά τότε αποθηκεύουμε το νέο βιβλίο στην βάση.

- Διαθεσιμότητα βιβλίου : Από την μεριά του client στέλνουμε απλά στον server το ISBN του βιβλίου μέσω ajax post request.
  Από την μεριά του server ελέγχουμε αρχικά ότι το ISBN αυτό υπάρχει στην βάση, στην συνέχεια βρισκουμε το id της βιβλιοθήκης και αν δεν υπάρχει ήδη το βιβλίο αυτό τότε το προσθέτουμε στον πίνακα booksinlibraries στην βάση.

- Ενημέρωση κατάστασης δανεισμού : Για να πάρουμε τον πίνακα με τα requests (και τις σχετικές πληροφορίες), στέλνουμε από την μεριά του client ένα απλό get request στον server. Από την μεριά του server βρίσκουμε αρχικά το id της βιβλιοθήκης που έκανε το request και μέσω αυτού εντοπίζουμε στον πίνακα borrowing όλα

τα βιβλία που έχουν δανειστεί χρήστες από αυτή την βιβλιοθήκη. Στην συνεχεια για κάθε τέτοιο βιβλίο παίρνουμε από την βάση το isbn και το status του, καθώς και το ονομα-email του χρήστη τα οποία επιστρέφουμε στον client σε μορφή json string.
Όσον αφορά το update κομμάτι από την μεριά του client βάζουμε ένα κουμπί δίπλα σε κάθε πεδίο του πίνακα που έχει στάτους requested ή returned και όταν το πατάει ο librarian στέλνεται στον server το borrowing_id του συγκεκριμένου βιβλίου μέσω PUT request. Τέλος από την μεριά του server βρίσκουμε στον πίνακα borrowing το βιβλίο με αυτό το borrowing_id και ανανεώνουμε το status του. Όταν το status ανανεώνεται σε successEnd τότε αλλάζουμε και στον πίνακα booksinlibraries το πεδίο 'available' του συγκεκριμένου βιβλίου σε true, για να μπορεί να δανειστεί κάποιος άλλος φοιτητής το βιβλίο.

- Προβολή ενεργών δανεισμών : Από την μεριά του client στέλνουμε απλά ένα get request στον server και λαμβάνουμε ένα json string με πληροφορίες για κάθε βιβλίο ως απάντηση, το οποίο προβάλλουμε στην σελίδα με πινακάκια html.
Από την μερία του server βρίσκουμε αρχικά το id βιβλιοθήκης και για κάθε βιβλίο που έχει δανειστεί από αυτή τη βιβλιοθήκη βρίσκουμε 1.το ISBN και τον τίτλο του από τους πίνακες booksinlibraries και books 2.όνομα,email,πανεπιστήμιο και τηλέφωνο του φοιτητή που το έχει δανειστεί από την πίνακα students και 3. το status του βιβλίου και υπολογίζουμε πόσες μέρες απομένουν μέχρι το τέλος του δανεισμού. Τέλος περνάμε όλες αυτές τις πληροφορίες σε ένα JsonObject και στην συνεχεια βάζουμε κάθε τέτοιο object σε ένα jsonArray το οποίο επιστρέφουμε τελικά στον client σε μορφή String.

4. Λειτουργίες Φοιτητή:

Τα servlets που χρειάζονται για αυτό το κομμάτι βρίσκονται στο φάκελο java/servlets/Student, αλλά χρησιμοποιούνται και άλλα κοινόχρηστα servlets που βρίσκονται στους φακέλους books και utilities (logout, ελεγχος για διπλα username κλπ).Από την μεριά του client, πέρα από το αρχείο

studentActions.js όλα τα υπόλοιπα είναι επίσης κοινα (με αυτά για τους librarians) .

Πως είναι υλοποιημένη η κάθε λειτουργικότητα :

- Register : Ακριβώς το ίδιο με το registration του librarian (υπάρχει μόνο ένα extra get request για να ελέγξουμε αν υπαρχει το academic id στην βάση).
- Login : Ακριβώς το ίδιο με το login του librarian. Τα ίδια ισχύουν και για τον έλεγχο του αν είναι ήδη συνδεδεμένος αλλά και για το logout.
- Η ανανέωση στοιχείων είναι πάνω κάτω ίδια διαδικασία με το register τόσο από την μερια του server αλλά και του client.

- Εύρεση Βιβλίων/Επέκταση API της 3ης άσκησης : Η διαδικασία εύρεσης βιβλίων είναι ίδια με αυτή του guest (χρησιμοποιούμε την ίδια φόρμα και συναρτήσεις-servlets)και περιγράφεται παραπάνω. Τώρα το πως έγινε η επέκταση του API : Στην 3η άσκηση βρίσκαμε τα βιβλία με το δοθέν genre και/ή publication year και τα αποθηκεύαμε σε ένα ArrayList το οποίο μετατρέπει αργότερα σε json string και το επιστρέφαμε στον client. Τώρα που υπάρχουν πιο πολλές επιλογές για τον χρήστη (author, title κλπ), κρατάμε στην άκρη το ArrayList που περιέχει τα βιβλία με το είδος/έτος και στην συνέχεια για κάθε άλλη κατηγορία που έχει δώσει ο χρήστης δημιουργούμε ένα ακόμη προσωρινό ArrayList για να κρατάμε τα βιβλία αυτής της κατηγορίας. Τέλος χρησιμοποιούμε την μέθοδο retainAll της ArrayList και διαγράφουμε από το Κύριο ArrayList όλα τα στοιχεία που υπάρχουν στο προσωρινό και δεν υπάρχουν στο κύριο. Ωστόσο για να λειτουργεί σωστά η μέθοδος retainAll υλοποιούμε την συνάρτηση equals στην κλάση books έτσι ώστε να γίνεται σωστή σύγκριση ανάμεσα στα book objects. Επιπλέον για κάθε βιβλίο εδώ υπάρχει ένα κουμπί που όταν πατηθεί στέλνεται ένα get request στον server και επιστρέφονται όλες οι κριτικές για το συγκεκριμένο βιβλίο από την βάση (κάτι που δεν υπάρχει στην εύρεση βιβλίων για τους guests).

- Εύρεση πιο κοντινής βιβλιοθήκης : Από την μεριά του server όταν πατηθεί ένα οποιοδήποτε κουμπί “show available libraries”, βρίσκουμε στην βάση το lat και lon του φοιτητή και το επιστρέφουμε στον client. Στην συνέχεια σε ένα άλλο servlet (άλλο request που λάβαμε από τον client), βρίσκουμε και

επιστρέφουμε από την βάση όλες τις απαραίτητες πληροφορίες για κάθε βιβλιοθήκη που έχει διαθέσιμο το βιβλίο που επέλεξε ο χρήστης.
Ο client από την μεριά του λαμβάνει τις παραπάνω πληροφορίες και 1. με τη χρήση του TrueWay Matrix API (και των lat/lon όλων τον βιβλιοθηκων και του φοιτητή) υπολογίζει τον χρόνο και την απόσταση για κάθε βιβλιοθήκη από τη διεύθυνση του φοιτητή. 2.Προσθέτει σε κάθε JsonObject (στις πληροφορίες κάθε βιβλιοθήκης) δύο νέα πεδία (χρόνο-απόσταση).3. Ταξινομεί τα objects με βάση την απόσταση και 4. Εμφανίζει τις πληροφορίες για κάθε βιβλιοθήκη στην html σελίδα μας.

- Δανεισμός βιβλίου : Όταν ο χρήστης επιλέξει την βιβλιοθήκη από την οποία θα δανειστεί το βιβλίο στέλνουμε από την μεριά του client ένα post request στον server με το bookcopy_id αυτού του βιβλίου. Από την μερία του server θέτουμε το βιβλίο ως μη διαθέσιμο από αυτή τη βιβλιοθήκη και προσθέτουμε ένα καινούριο entry στον πίνακα borrowing με τα στοιχεία αυτού του δανεισμού.Επιπλέον αυξάνουμε κατά ένα το πεδίο borrow_count για τον συγκεκριμένο φοιτητή το οποίο χρειαζόμαστε στο bonus ερώτημα με τα giveaways.
- Επιστροφή βιβλίου : Όταν ο χρήστης πατήσει το “View Borrowings” στο navbar του εμφανίζονται όλα τα βιβλία που έχει δανειστεί (και αυτά που είχε δανειστεί στο παρελθόν). Για αυτά που έχει δανειστεί στο παρόν (είναι σε κατάσταση borrowed) υπάρχει ένα κουμπί ‘return’ το οποίο όταν πατηθεί στέλνεται ένα put request στον server με το borrowing_id του συγκεκριμένου βιβλίο και ανανεώνεται η κατάσταση του στον πίνακα borrowing από borrowed σε returned.

- Ειδοποίηση για τέλος δανεισμού : Κάθε φορά που κάνει login ο χρήστης (ή reload το page όσο είναι logged-in) τρέχουμε μια συνάρτηση από την μεριά του client η οποία στέλνει ένα απλό get request στον server. Στο server side βρισκουμε το Id του συνδεδεμένου χρήστη και παίρνουμε από τον πίνακα borrowing όλα τα βιβλία που έχει δανειστεί αυτός ο χρήστης και είναι σε κατάσταση borrowed. Για κάθε τέτοιο βιβλίο υπολογίζουμε πόσες μέρες δανεισμού απομένουν και αν είναι <=3 ή < 0 (δηλαδή έχει ξεπεράσει το όριο) επιστρέφουμε το κατάλληλο μήνυμα στον

client. (Όταν ο χρήστης κάνει logout ή κάνει return τα βιβλία αυτά προφανώς κρύβουμε το μήνυμα).

- Καταχώρηση Κριτικής : Πάλι όταν ο χρήστης πατήσει το "View Borrowings" και εμφανιστούν όλα τα βιβλία που έχει δανειστεί, για αυτά που είναι σε κατάσταση successEnd υπάρχει ένα κουμπί που όταν πατηθεί εμφανίζεται μία φόρμα στην οποία ο χρήστης μπορεί να γράψει την κριτική. Όταν την υποβάλλει την στέλνουμε στον server μέσω xml post request μαζί με το isbn του συγκεκριμένου βιβλίου και από την μερία του server προσθέτουμε την κριτική στον πίνακα reviews. Επιπλέον αυξάνουμε κατά ένα το πεδίο borrow_count για τον συγκεκριμένο φοιτητή το οποίο χρειαζόμαστε στο bonus ερώτημα με τα giveaways.

-Bonus ερώτημα (Giveaway) :

Από την μερια του client : Στην σελίδα του administrator υπάρχει ένα κουμπί που όταν πατηθεί στέλνεται ένα post request στον server και εκεί τρέχει το giveaway.
Για την προβολή των αποτελεσμάτων στους χρήστες, κάθε φορά που κάνει ένας χρήστης login (ή μπαίνει στην σελίδα για δεύτερη φορά χωρίς να κάνει logout) στέλνεται ένα get request στον server και αν υπάρχει στον πίνακα adminmessages τέτοιο μήνυμα τότε το επιστρέφει ο server και το τυπώνουμε στην αρχική σελίδα.
Από την μερία του server για την υλοποίηση του αλγορίθμου για το giveaway: Παίρνουμε από τη βάση τα ονόματα και τον αριθμό των συμμετοχών όλων των φοιτητών που έχουν δανειστεί τουλαχιστον ένα βιβλίο (έχουν τουλάχιστον μια συμμετοχή). Έχουμε δημιουργήσει μία κλάση Competitor όπου για κάθε χρήστη κρατάμε το όνομα του και τον αριθμό των συμμετοχών του. Επιπλέον υπάρχει και ένα ArrayList entries στο οποίο θα αποθηκεύουμε τόσα στοιχεία όσα και ο αριθμός των συμμετοχών του χρήστη (τα στοιχεία που αποθηκεύουμε μέσα ειναι αριθμοι ως εξής : Αν ο πρώτος χρήστης έχει 2 συμμετοχές τοτε οι αριθμοι που θα βάλουμε στο entries του θα ειναι οι 1 κ 2, Αν ο δεύτερος

χρήστης έχει 3 συμμετοχες τότε στο entries του θα βάλουμε τους αριθμούς 3,4 κ 5. Και ουτω καθεξης για τους υπόλοιπους χρήστες)
Για να βγάλουμε νικητή δημιουργούμε ένα τυχαίο αριθμό από το 1 μέχρι τον συνολικό αριθμό των συμμετοχών και ελέγχουμε ποιος από τους competitors μας έχει αυτόν τον αριθμο στο entries του. Με αυτό το τρόπο οι χρήστες με τις περισσότερες συμμετοχές έχουν και μεγαλύτερη πιθανότητα να κερδίσουν.
Για την ανάδειξη του δεύτερου νικητή κάνουμε το ίδιο απλά αυτή τη φορά έχουμε αφαιρέσει τον πρώτο νικητή από την λίστα των competitors. Τέλος τα αποτελέσματα της κλήρωσης αποθηκεύονται στον πίνακα adminmessages στην βάση για να τα προβάλουμε αργότερα στη σελίδα των χρηστών.