

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ



ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Ακαδημαϊκό έτος 2022-2023, 6ο εξάμηνο

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ – ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ

Ομάδα Project 8

Μαρία Γρατσία (*el20082*)

Παναγιώτα Μπρέζα (*el20812*)

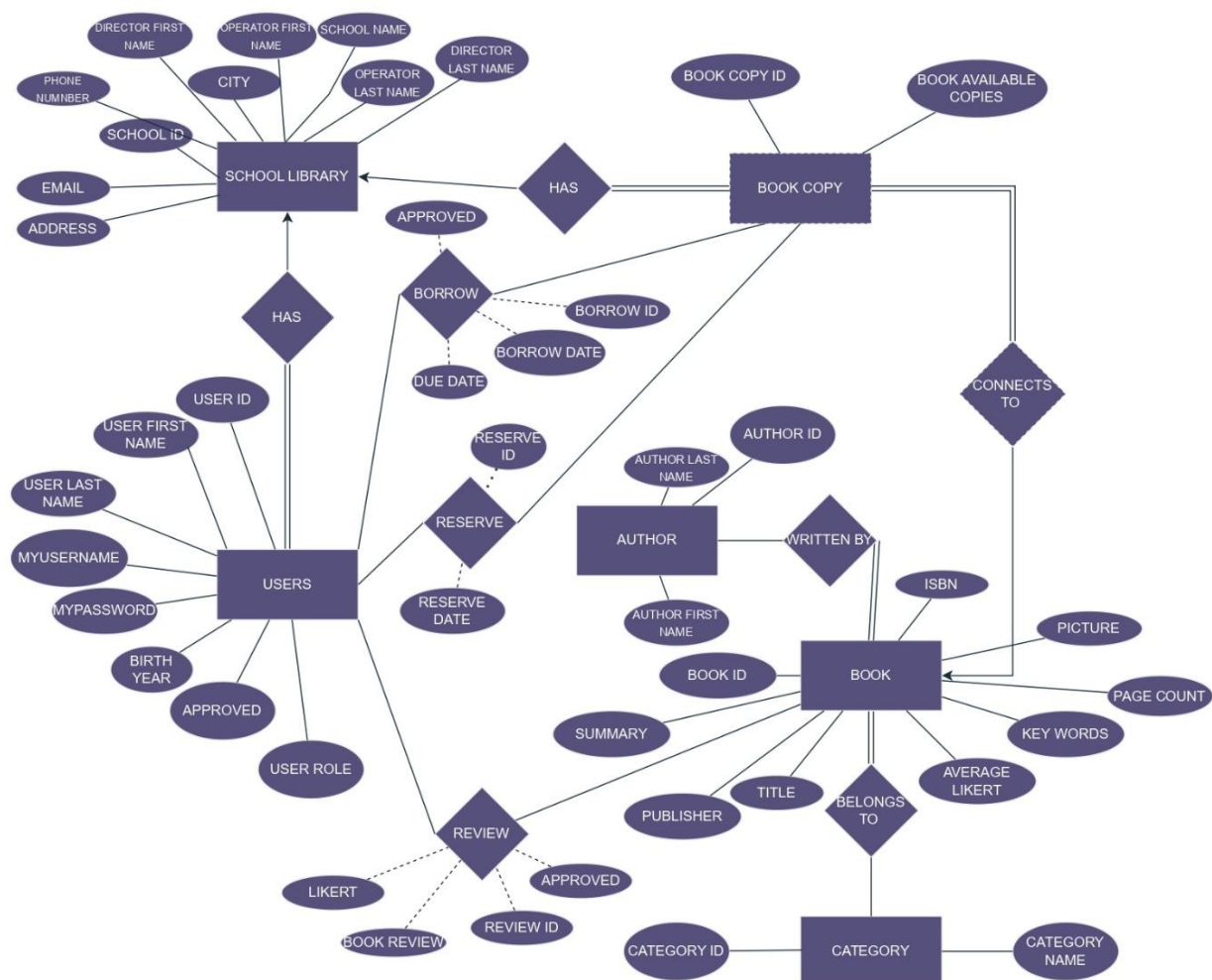
Περιεχόμενα

1. Entity-Relationship diagram
2. Relational Schema
 - 2.1. DDL & DML
 - 2.2. Constraints, Keys
 - 2.3. Data
 - 2.4. Indexes
3. Queries
4. User Manual

Στην εργασία αυτή προχωρήσαμε στην υλοποίηση ενός συστήματος αποθήκευσης και διαχείρισης πληροφοριών για τη λειτουργία σχολικών βιβλιοθηκών σε δημόσια σχολεία.

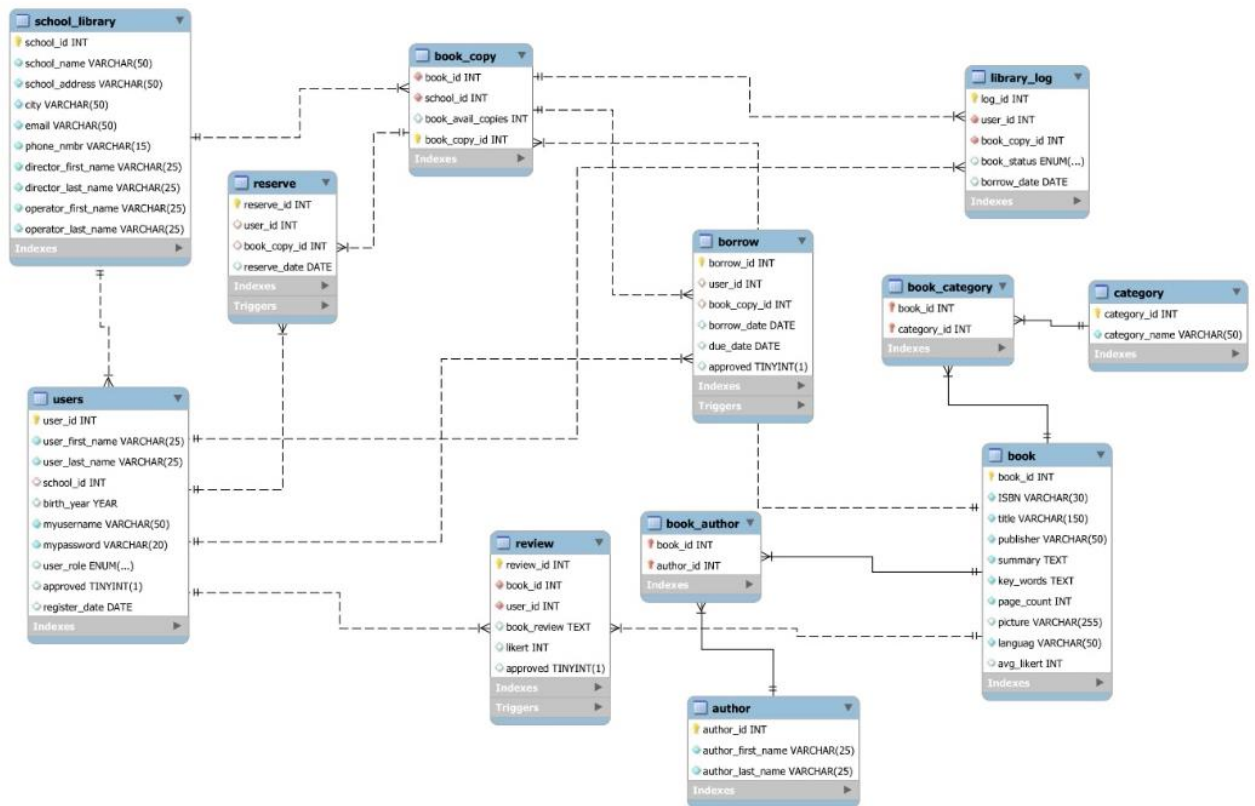
1. Entity-Relationship Diagram

Πρώτο βήμα, η κατασκευή του διαγράμματος οντοτήτων-συσχετίσεων. Το τελικό διάγραμμα που προέκυψε είναι το ακόλουθο:



2. Relational Schema

Έπειτα προχωρήσαμε στη δημιουργία του σχεσιακού διαγράμματος με αποτέλεσμα:



2.1 Data Definition Language (DDL) - Data Manipulation Language (DML)

Έχοντας αυτά τα εργαλεία στα χέρια μας, μπορούμε να προχωρήσουμε στην ανάπτυξη της βάσης μας και συγκεκριμένα στο Data Definition Language (DDL) και Data Manipulation Language (DML). Παρακάτω παραθέτουμε τη δημιουργία των πινάκων και της βάσης δεδομένων:

Create Database AbsoluteMinds -Create table school library

```
1 • CREATE DATABASE IF NOT EXISTS AbsoluteMinds;
2 • USE Absoluteminds;
3
4 #School Library Table
5 • create table if not exists school_library (
6   school_id int unsigned not null auto_increment,
7   school_name varchar(50) not null unique,
8   school_address varchar(50) not null unique,
9   city varchar(50) not null,
10  email varchar(50) not null unique,
11  phone_nmbr varchar(15) not null unique,
12  director_first_name varchar(25) not null,
13  director_last_name varchar(25) not null,
14  operator_first_name varchar(25) not null,
15  operator_last_name varchar(25) not null,
16  primary key (school_id)
17 );
18
```

Create table book, category, author

```
19 #Book Table
20 • create table if not exists book (
21     book_id int unsigned not null auto_increment,
22     ISBN varchar(30) not null unique,
23     title varchar(150) not null,
24     publisher varchar(50) not null,
25     summary text not null,
26     key_words text not null,
27     page_count int not null,
28     picture varchar(255),
29     languag varchar(50) not null default 'English',
30     avg_likert int default null check (avg_likert between 1 and 5),
31     primary key (book_id)
32 );
33
34 #Category Table
35 • create table if not exists category (
36     category_id int unsigned not null auto_increment,
37     category_name varchar(50) not null unique,
38     primary key (category_id)
39 );
40
41 #Author Table
42 • create table if not exists author (
43     author_id int unsigned not null auto_increment,
44     author_first_name varchar(25) not null,
45     author_last_name varchar(25) not null,
46     primary key (author_id)
47 );
```

Create table book_category, book_author

```
48 #Book_category Table
49 • create table if not exists book_category (
50     book_id int unsigned not null,
51     category_id int unsigned not null,
52     primary key (book_id, category_id),
53     constraint fk_book_category_book
54         foreign key (book_id)
55             references book (book_id)
56             on update cascade
57             on delete cascade,
58     constraint fk_book_category_category
59         foreign key (category_id)
60             references category (category_id)
61             on update cascade
62             on delete cascade
63 );
64
65 #Book_author Table
66 • create table if not exists book_author (
67     book_id int unsigned not null,
68     author_id int unsigned not null,
69     primary key (book_id, author_id),
70     constraint fk_book_author_book
71         foreign key (book_id)
72             references book (book_id)
73             on update cascade
74             on delete cascade,
75     constraint fk_book_author_author
76         foreign key (author_id)
77             references author (author_id)
78             on delete cascade
79             on update cascade
80 );
81
```

Create table book_copy

```
81
82 #Book_copy Table
83 • create table if not exists book_copy (
84     book_id int unsigned not null,
85     school_id int unsigned not null,
86     book_avail_copies int unsigned,
87     book_copy_id int unsigned not null auto_increment,
88     primary key (book_copy_id),
89     constraint fk_book_copy_book
90         foreign key (book_id)
91         references book (book_id)
92         on delete cascade
93         on update cascade,
94     constraint fk_book_copy_school_library
95         foreign key (school_id)
96         references school_library (school_id)
97         on delete cascade
98         on update cascade
99 );
100
101
```

Create user table

```
101
102 #User Table
103 # S = Student, T = Teacher, O = Library Operator, M = System Manager
104 • create table if not exists users (
105     user_id int unsigned not null auto_increment,
106     user_first_name varchar(25) not null,
107     user_last_name varchar(25) not null,
108     school_id int unsigned,
109     birth_year year,
110     myusername varchar(50) not null unique,
111     mypassword varchar(20) not null constraint length check (char_length(mypassword) between 3 and 15),
112     user_role enum ('S', 'T', 'O', 'M'),
113     approved bool default false,
114     register_date date default (current_date),
115     primary key (user_id),
116     constraint fk_users_school_library
117         foreign key (school_id)
118         references school_library (school_id)
119         on delete cascade
120         on update cascade
121 );
```

Create borrow table

```
122
123 #Borrow Table
124 • create table if not exists borrow (
125     borrow_id int unsigned not null auto_increment,
126     user_id int unsigned,
127     book_copy_id int unsigned,
128     borrow_date date default (current_date),
129     due_date date default (date_add(borrow_date, interval 7 DAY)),
130     approved bool default false,
131     primary key (borrow_id),
132     constraint fk_borrow_users
133         foreign key (user_id)
134         references users (user_id)
135         on delete cascade
136         on update cascade,
137     constraint fk_borrow_book_copy
138         foreign key (book_copy_id)
139         references book_copy (book_copy_id)
140         on delete cascade
141         on update cascade
142 );
```

Create reserve table

```
144 #Reserve Table
145 • create table if not exists reserve (
146     reserve_id int unsigned not null auto_increment,
147     user_id int unsigned,
148     book_copy_id int unsigned,
149     reserve_date date default (current_date),
150     primary key (reserve_id),
151     constraint fk_reserve_users
152         foreign key (user_id)
153         references users (user_id)
154         on delete cascade
155         on update cascade,
156     constraint fk_reserve_book_copy
157         foreign key (book_copy_id)
158         references book_copy (book_copy_id)
159         on delete cascade
160         on update cascade
161 );
```

Create library log table

```
171 #library_log Table
172 • create table if not exists library_log (
173     log_id int unsigned not null auto_increment,
174     user_id int unsigned not null,
175     book_copy_id int unsigned not null,
176     book_status enum ('Borrowed', 'Returned', 'Reserved'),
177     borrow_date date default (current_date),
178     primary key (log_id),
179     constraint fk_log_users
180         foreign key (user_id)
181         references users (user_id)
182         on delete cascade
183         on update cascade,
184     constraint fk_log_book_copy
185         foreign key (book_copy_id)
186         references book_copy (book_copy_id)
187         on delete cascade
188         on update cascade
189 );
```

Create review table

```
246
247 • #Review Table
248 • create table if not exists review (
249     review_id int unsigned not null auto_increment,
250     book_id int unsigned not null,
251     user_id int unsigned not null,
252     book_review text default null,
253     likert int default null check (likert between 1 and 5),
254     approved bool default false,
255     primary key (review_id),
256     constraint fk_review_users
257         foreign key (user_id)
258         references users (user_id)
259         on delete cascade,
260     constraint fk_review_book
261         foreign key (book_id)
262         references book (book_id)
263         on delete cascade
264 );
```

Για λόγους λειτουργικότητας της βάσης προσθέσαμε trigger και events:

```

191
192 #Triggers for update library_log and book availability
193 delimiter $$
194 ● create trigger after_borrow_before_approved
195 after insert on borrow
196 for each row
197 begin
198 declare new_number int unsigned;
199 set new_number = (select book_avail_copies from book_copy where book_copy_id = new.book_copy_id) - 1;
200 insert into library_log(user_id, book_copy_id, book_status, borrow_date)
201 values (new.user_id, new.book_copy_id, 'Reserved', new.borrow_date);
202 update book_copy
203 set book_avail_copies = new_number
204 where book_copy_id = new.book_copy_id;
205 end$$
206
207 ● create trigger new_log_after_reserve
208 after insert on reserve
209 for each row
210 begin
211 insert into library_log(user_id, book_copy_id, book_status)
212 values (new.user_id, new.book_copy_id, 'Reserved');
213 end$$
214
215 ● create trigger after_approved
216 after update on borrow
217 for each row
218 begin
219 update library_log
220 set book_status = 'Borrowed', borrow_date = current_date
221 where old.book_copy_id = book_copy_id and old.user_id = user_id;
222 end$$
223
224 ● create trigger after_return
225 before delete on borrow
226 for each row
227 begin
228 declare new_number int unsigned;
229 set new_number = (select book_avail_copies from book_copy where book_copy_id = old.book_copy_id) + 1;
230 update library_log
231 set book_status = 'Returned'
232 where old.book_copy_id = book_copy_id;
233 update book_copy
234 set book_avail_copies = new_number
235 where old.book_copy_id = book_copy_id;
236 end$$
237
238
239 ● create trigger delete_from_log_after_cancel_reserve
240 before delete on reserve
241 for each row
242 begin
243 delete from library_log
244 where old.book_copy_id = book_copy_id and old.user_id = user_id;
245 end$$
246
247 delimiter ;
248
249 ---
250
251 #Event to delete reserves after 7 days of their creation
252 ● set global event_scheduler = on;
253 ● create event if not exists delete_reserve_after_7_days
254 on schedule every 1 day
255 starts (select date(current_timestamp())) + interval 1 day
256 do
257 delete from reserve where reserve_date < date_sub(curdate(), interval 7 day);
258
259

```

Ακόμα για λόγους διευκόλυνσης προχωρήσαμε στη διαφοροποίηση των δεδομένων με μια χαρακτηριστική χιλιάδα.

```

278
279 • #Special ID number for each data
280 alter table school_library auto_increment = 1000;
281 • alter table book auto_increment = 2000;
282 • alter table author auto_increment = 3000;
283 • alter table category auto_increment = 4000;
284 • alter table users auto_increment = 5000;
285 • alter table book_copy auto_increment = 6000;
286

```

2.2 Constraints, Keys, values integrity

Έπειτα θέσαμε κάποιους απαραίτητους περιορισμούς καθώς και τα κλειδιά όπως παρατηρούμε παραπάνω. Συγκεκριμένα έχουμε τους ακόλουθους περιορισμούς με τη σειρά:

- avg_likert = τιμές μεταξύ 1 – 5
- category_name = unique
- mypassword length = τιμές μεταξύ 3 – 15
- κάθε σχολείο έχει έναν operator
- κάθε βάση δεδομένων έχει μόνο έναν manager
- κάθε κράτηση διαγράφεται μετά το πέρας 7 ημερών
- ένας μαθητής μπορεί να δανειστεί/κάνει κράτηση έως 2 βιβλία
- ένας καθηγητής μπορεί να δανειστεί/κάνει κράτηση έως 1 βιβλίο
- ένας μαθητής/καθηγητής δε μπορεί να δανειστεί άμα έχει καθυστερήσει να επιστρέψει ένα βιβλίο
- ένας χειριστής/διαχειριστής δε μπορεί να κάνει κράτηση/δανεισμό/αξιολόγηση

2.3 Data for database

Στο αρχείο insert_data.sql βάλαμε τα απαραίτητα δεδομένα για όλες τις οντότητες. Συγκεκριμένα προσθέσαμε:

- 8 σχολεία
- 105 βιβλία
- 12 κατηγορίες
- 60 συγγραφείς
- 352 φυσικά αντίτυπα των βιβλίων (διαχωρισμένα ανάμεσα στα 8 σχολεία)
- 80 user όπου έχουμε 8 operators, 47 students, 25 καθηγητές
- 1 user - Library manager
- Προσθέσαμε παρελθοντικούς δανεισμούς στο library_log
- 8 δανεισμοί που έχουν καθυστερήσεις την επιστροφή τους
- 50 ενεργούς δανεισμούς (διαμοιρασμένους ανάμεσα στα 8 σχολεία)
- 12 δανεισμοί που περιμένουν την έγκριση του διαχειριστή
- 46 κρατήσεις
- 5 αξιολογήσεις

2.4 Indexes

Με στόχο τη βελτίωση της αναζήτησης και της απόδοσης των queries, δημιουργήσαμε κάποια indexes. Επιλέξαμε τα indexes ώστε να βελτιώσουν την απόδοση στα πιο χρονοβόρα και στα πιο συχνά queries:

```
286
287 #INDEXES
288
289 • create index idx_due_date on borrow (due_date);
290 • create index idx_school_name on school_library (school_name);
291 • create index idx_username on users(myusername);
292 • create index idx_user_first_name on users(user_first_name);
293 • create index idx_user_last_name on users(user_last_name);
294 • create index idx_ISBN on book(ISBN);
295 • create index idx_borrow_date on borrow(borrow_date);
296
```

3. QUERIES

Υλοποιήσαμε τα ζητούμενα queries και έπειτα τα συνδέσαμε με το UI, με τέτοιο τρόπο ώστε ένας μη γνώστης της SQL να μπορεί να χρησιμοποιήσει τη βάση μας. Μπορούμε να δούμε τη εκτέλεση των queries στο επίπεδο UI.

Για να διευκολύνουμε τα queries φτιάξαμε τα ακόλουθα views:

```
297
298 #VIEWS
299
300 #Late_returns
301 • create view late_returns as
302   select user_id, book_copy_id, due_date, datediff(current_date, due_date) as days_of_delay
303   from borrow
304   where datediff(current_date, due_date) > 0;
305
306 #Total borrows ever
307 • create view borrows_history as
308   select book_copy_id, user_id, borrow_date
309   from library_log
310   where book_status = 'Returned' or book_status = 'Borrowed';
311
312 #for 3.1.5 Operators and borrowed books
313 • create view operators_and_borrowed_books as
314   select any_value(operator_first_name) as operator_first_name, any_value(operator_last_name) as operator_last_name, count(book_copy_id) as borrowed_books, any_value(user_id) as user_id
315   from (( users
316     inner join school_library using (school_id)
317     inner join borrows_history using (user_id) )
318     where date_sub(borrow_date, interval 1 year)
319     GROUP BY (school_id);
320
321 # For 3.1.6
322 • create view book_category_name as
323   select book_id, category_id, category_name from
324   book_category inner join category using(category_id);
```

3.1 Διαχειριστής (Manager)

3.2 Χειριστής (Operator)

3.3 Χρήστης (User)

4. User Manual - Εγχειρίδιο Χρήστη για την Εφαρμογή

GITHUB LINK: <https://github.com/mariagratsia/libraryDB2023/tree/main>

Για την κατασκευή της εφαρμογής χρησιμοποιήσαμε Python (flask), HTML, λίγη CSS για την εικόνα.

Λήψη των απαραίτητων βιβλιοθηκών και λογισμικών

Λογισμικά

- <https://www.python.org/downloads/>
- <https://dev.mysql.com/downloads/workbench/>
- <https://dev.mysql.com/downloads/mysql/>

- Python editor της επιλογής μας (στην υλοποίηση μας χρησιμοποιήθηκε το visual studio: <https://code.visualstudio.com/download>)

Requirements

- Κατεβάζουμε το requirements.txt από το Github
- Ανοίγουμε το command prompt
- Πηγαίνουμε στο directory όπου έχει αποθηκευτεί το αρχείο requirements.txt (π.χ. αν είναι στο Desktop → `cd C:\Users\YourUsername\Desktop`)
- Εκτελούμε την ακόλουθη εντολή: `pip install -r requirements.txt`

Λήψη των στοιχείων

Κατεβάζουμε τα αρχεία SQL Code και User Interface από το σύνδεσμο του Github με ιδιαίτερη προσοχή ώστε να παραμείνουν οι φάκελοι με τη σωστή διαρρύθμιση.

Κατασκευή βάσης Δεδομένων

1. Δημιουργία βάσης δεδομένων μέσω του MySQL Workbench
2. Εισαγωγή δεδομένων μέσω του MySQL Workbench

Αλλαγές στη python

Εισαγωγή των δικών μας credentials στο αρχείο `__init__.py`

```
app.config["MYSQL_USER"] = '***'  
app.config["MYSQL_PASSWORD"] = '***'  
app.config["MYSQL_DB"] = 'absoluteminds'  
app.config["MYSQL_HOST"] = 'localhost'  
app.config["SECRET_KEY"] = '***'  
app.config["WTF_CSRF_SECRET_KEY"] = ''
```

Σημείωση

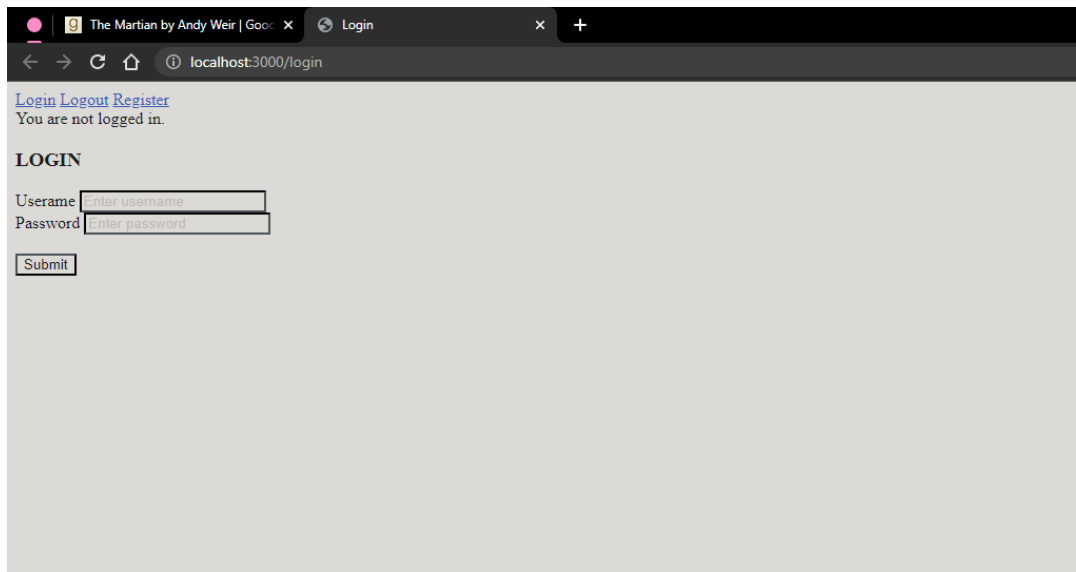
Έχουμε θέσει το port = 3000 στο αρχείο `run.py` σε περίπτωση conflict μπορεί να χρειαστεί να το αλλάξουμε.

Για να ξεκινήσουμε το UI αφού αποθηκεύσουμε όλα τα αρχεία τρέχουμε το `run.py`, το οποίο μας εμφανίζει ένα σύνδεσμο ή αν δεν εμφανιστεί αναζητούμε εμείς στο browser μας το localhost

Προσοχή! Οι φόρμες συμπλήρωσης είναι ιδιαίτερα case sensitive οπότε απαιτείται προσοχή στα πεζά και στα κεφαλαία καθώς και στα κενά έπειτα από το τέλος της λέξης

Για παράδειγμα το username 'bestdetective' και 'bestdetective' θεωρούνται διαφορετικά.

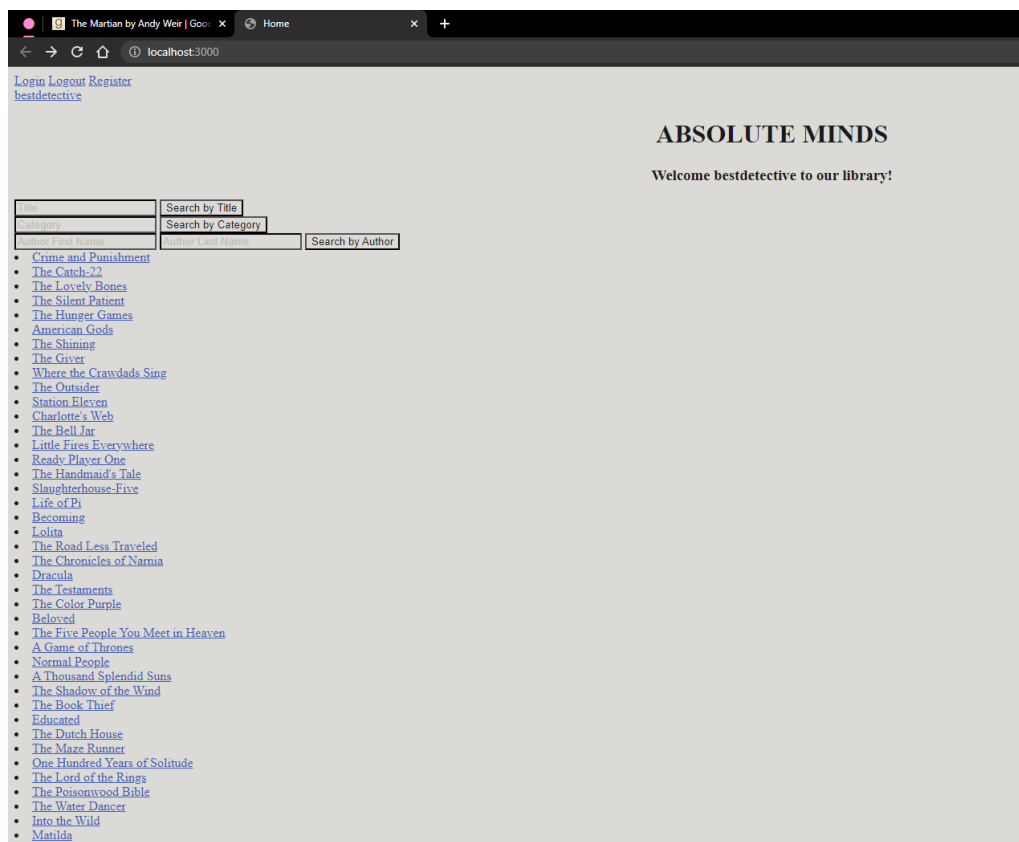
Για να ξεκινήσουμε το UI αφού αποθηκεύσουμε όλα τα αρχεία τρέχουμε το `run.py`, το οποίο μας εμφανίζει ένα σύνδεσμο. Αν τον ανοίξουμε μας εμφανίζει το login page:



Από εδώ μπορούμε να κάνουμε login.

- Παράδειγμα σύνδεσης για User (Student, Teacher) : Username = bestdetective Password = sherlocked

Τώρα είμαστε στο home page όπου εμφανίζονται όλα τα βιβλία που υπάρχουν στη βιβλιοθήκη του χρήστη.




Μπορούμε να ανοίξουμε ένα βιβλίο και να δούμε τις πληροφορίες του, να δημιουργήσουμε ένα αίτημα δανεισμού (το οποίο θα γίνει κράτηση αν δε υπάρχουν διαθέσιμα αντίτυπα) και δημιουργία αξιολόγησης:

← → ↻ 🏠 📄 localhost:3000/book_details/2039

[Login](#) [Logout](#) [Register](#)

Book Details



Title: The Martian
ISBN: 9780062315007
Publisher: Broadway Books
Summary: The Martian is a science fiction novel by Andy Weir published in 2011.
Keywords: survival, space exploration, humor, resilience
Page Count: 369
Language: English
Average Likert Rating: None
Categories:

- Thriller

Authors:

- Ryan Ward

[Login](#) [Logout](#) [Register](#)

Borrow Request Successful! Waiting for approval...

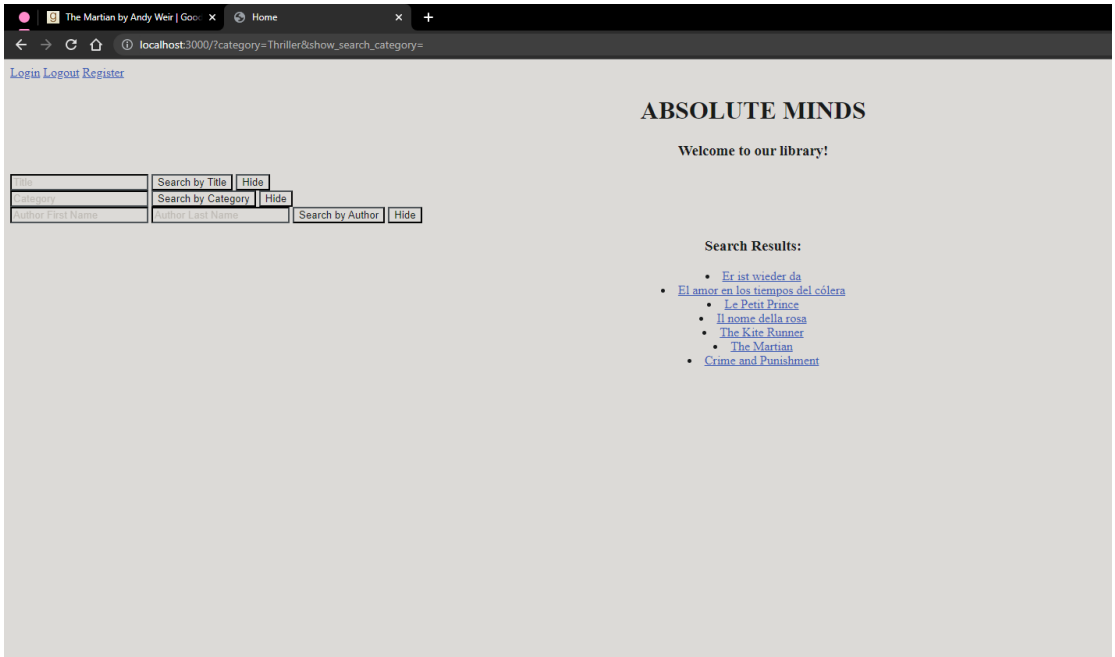
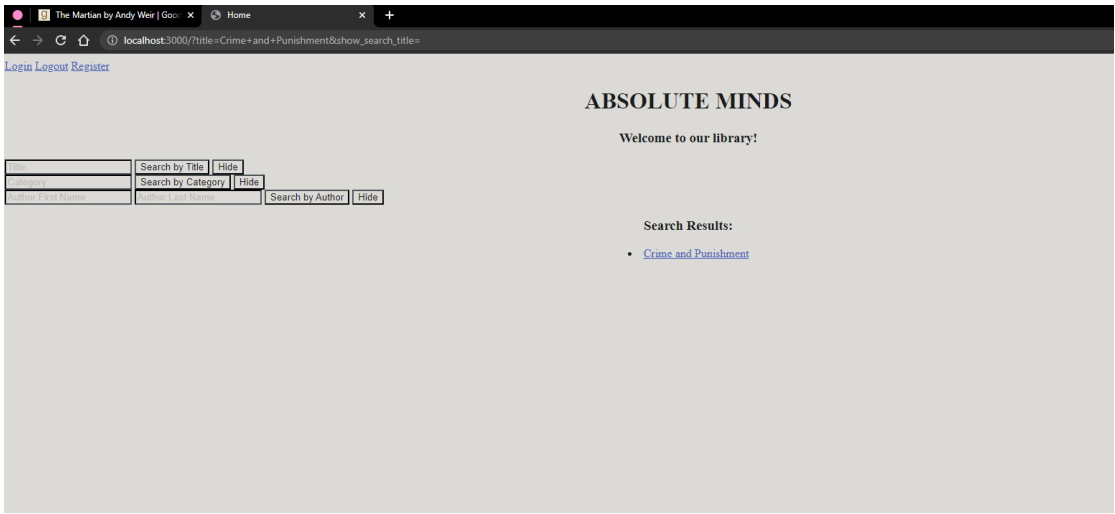
[Return to home page](#)

[Login](#) [Logout](#) [Register](#)
[Return to home page](#)

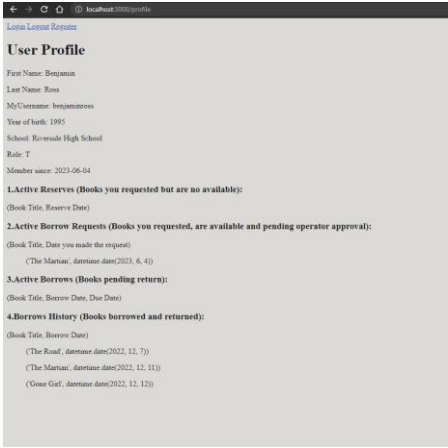
ADD REVIEW

Review
Likert

Μπορούμε να κάνουμε αναζήτηση με βάση το τίτλο, την κατηγορία και το συγγραφέα και να μας επιστρέψει αντίστοιχα τα βιβλία και μετά με το κουμπί **HIDE** μπορούμε να επιστρέψουμε στην προηγούμενη σελίδα:



Καθώς και να δούμε το profil μας:



- Παράδειγμα σύνδεσης για Operator : Username =karenlee, Password = admin123
Παρατηρούμε ότι μας εμφανίζονται και οι έξτρα επιλογές για το χειριστή σύμφωνα με τα ζητούμενα:

[Login](#) [Logout](#) [Register](#)
[karenlee](#)
Logged in successfully

ABSOLUTE MINDS

Welcome operator karenlee!

[Late Returns](#) [Reviews](#)

[Approve](#)

[Return](#) [Library](#) [Log](#)

Title	Search by Title		
Category	Search by Category		
Author First Name	Author Last Name	Search by Author	
Available Copies	Search by Available Copies		

Search Results:

No search results found.

School Books:

- [Ich bin dann mal weg: Meine Reise auf dem Jakobsweg](#)
 - [El amor en los tiempos del cólera](#)
 - [The Picture of Dorian Gray](#)
 - [Station Eleven](#)
 - [The Handmaid's Tale](#)
 - [Siddhartha](#)
 - [Fahrenheit 451](#)
 - [Il nome della rosa](#)
 - [American Gods](#)
 - [The Road Less Traveled](#)
 - [Er ist wieder da](#)
 - [1984](#)
 - [Brave New World](#)
 - [Crime and Punishment](#)
 - [Beloved](#)
- [Das Parfum: Die Geschichte eines Mörders](#)

- Παράδειγμα σύνδεσης για Manager : Username =Admin Password = 1234

← → ↻ 🏠 ⓘ localhost:3000/manager

[Login](#) [Logout](#) [Register](#)
[Admin](#)
Logged in successfully

ABSOLUTE MINDS

Welcome manager!

[Options](#)

Registered Schools

[Fairview Middle School](#)

[Maplewood Elementary](#)

[Meadowbrook Elementary](#)

[Oakwood Academy](#)

[Riverside High School](#)

[Springfield Middle School](#)

[Sunrise High School](#)

[Westwood High School](#)

- ΠΑΡΑΔΟΧΕΣ:

- Για να εγγραφεί ένας χρήστης στην εφαρμογή χρειάζεται τον μοναδικό κωδικό του σχολείου του, τον οποίο υποθέσαμε ότι λαμβάνει από τον χειριστή.
- Για να δανειστεί ένα βιβλίο ένας χρήστης πρέπει πρώτα να κάνει borrow request μέσω της εφαρμογής και έτσι το βιβλίο κρατείται (reserve). Ο χειριστής κάνει approve το borrow request κατά την φυσική παραλαβή του βιβλίου από τον χρήστη.
- Σε περίπτωση που ένα βιβλίο δεν είναι διαθέσιμο, δίνεται στον χρήστη η επιλογή να το κάνει κράτηση (reserve) και να περιμένει την επιστροφή κάποιου αντίτυπου.
- Επιστροφές βιβλίων μπορεί να εκτελέσει μόνο ο χειριστής, με την εισαγωγή username και ISBN στο σύστημα.
- Κατά την επιστροφή ενός αντίτυπου, ελέγχονται αυτόματα οι κρατήσεις που αφορούν βιβλία χωρίς διαθεσιμότητα, και αν κάποια μπορεί να ικανοποιηθεί, μετατρέπεται αυτόματα σε borrow request και περιμένει την φυσική παραλαβή του αντίτυπου από την βιβλιοθήκη ώστε να γίνει approved.