Referat proiect EWD

Cerințe și mod de funcționare al aplicației

Ca proiect pentru materia EWD am ales să realizez prima cerință, care era construirea unui site web și respectarea unui număr minim de cerințe. Tema care am ales-o a fost un tasklist ce folosește o bază de date pentru a stoca informațiile introduse.

Pentru a putea salva și a accesa informațiile introduse, utilizatorul este obligat să iși creeze un cont înainte de a putea accesa tasklistul. Dacă acesta încearcă să acceseze site-ul fără să se fi logat, acesta va fi redirecționat către pagina de login. Când se înregistrează, utilizatorul trebuie să introducă un nume de minimum 3 caractere si o parola de minimum 6 caractere, altfel nu vor fi trimise informațiile introduse. După ce se loghează folosind numele si parola, utilizatorul va fi redirecționat către tasklist. În partea de sus stângă a tasklistului va apărea numele utilizatorului cu o săgeata în jos. Dacă se va pune pointer-ul pe acesta, va apărea butonul de log out, prin care utilizatorul se va deloga (se va încheia sesiunea) și va fi redirecționat înapoi la pagina de login. Tasklistul este însă împărțit in 4 părți:

1. În partea de sus se află două input-uri, cel de mai sus fiind pentru adăugarea task-ului, iar cel de dedesubt fiind pentru adăugarea categoriei. Pentru a le folosi, utilizatorul va introduce task-ul (sau numele noii categorii), iar apoi va apăsa pe butonul de „Add”, respectiv „Add Category”. Cutia de selecție din dreapta input-ului pentru adăugarea task-ului este pentru a selecta o categorie. Dacă utilizatorul dorește să introducă task-ul într-o categorie specifică, acesta va folosi cutia de selecție pentru a specifica categoria.
2. Partea de „Display category”. Aceasta conține o cutie de selecție care listează toate categoriile existente în baza de date pentru utilizator. Lângă cutia de selecție se află două butoane, „Delete Category” și „Edit Category”, care va șterge categoria selectată, respectiv va lăsa utilizatorul sa editeze numele categoriei selectate.
3. Lista „To do”, în care se vor afișa task-urile nou adăugate sau deja existente care nu au fost terminate (mutate la lista „Done”). Pentru fiecare task există un checkbox și două butoane prin care ele pot fi selectate sau respectiv, editate sau șterse. Deasupra listei de task-uri se află butoane care interacționează doar cu checkbox-urile de la task-uri. De la stânga la dreapta, ele pot să mute, șteargă, sau să schimbe categoria elementelor selectate (Cutia de selecție este pentru alegerea categoriei în care utilizatorul vrea să schimbe elementele selectate). De asemenea, există un buton „Select All” care se poate folosi la selectarea tuturor elementelor din listă.
4. Lista „Done”, în care se pot muta task-urile terminate. Aceasta funcționează la fel ca și lista anterioară.

În mare parte, tasklist-ul funcționează prin Javascript și PHP. Schimbările în timp real sunt făcute cu Javascript, iar operațiile de stocare în baza de date MYSQL sunt făcute prin PHP. De asemenea, când utilizatorul intră pe tasklist, prima dată listele și opțiunile de la cutiile de select sunt construite prin PHP, care accesează informațiile din baza de date și le prelucrează, apoi le adaugă bazat pe ce task-uri sau categorii găsește că aparțin id-ului utilizatorului logat. Pentru fiecare task adăugat, numele acestuia și id-ul categoriei în care a fost introdus vor fi trimise cu ajutorul AJAX la „addtask.php”, unde vor fi prelucrate și adăugate cu id-ul utilizatorului a cărui aparține acest task și statusul acestui task (dacă e gata sau nu) în baza de date, în tabelul „tasks”. Pentru fiecare categorie adaugată, numele acesteia va fi trimis cu ajutorul AJAX la „addcategory.php”, de unde va fi adaugată împreună cu id-ul utilizatorului în baza de date, în tabelul „categories”. De asemenea, când un utilizator se înregistrează, numele, parola și emailul vor fi de asemenea stocate în baza de date, în tabelul „users”.

Descrierea funcționalităților aplicației

Conectarea la și crearea bazei de date (daca aceasta nu există) împreună cu tabelele sale se face în fișierul „connect.php” .

Înregistrarea se face accesând „register.php”, care este o pagină cu un formular ce are trei inputuri. Unul pentru nume, unul pentru parolă și unul pentru email. În site este un mic script Javascript scris care la trimiterea formularului verifică dacă numele este de minim 3 caractere și dacă parola este de minim 6 caractere. Dacă nu sunt respectate aceste cerințe, formularul nu este trimis. Aceste date sunt trimise către „registerform.php”, unde se verifică dacă există deja numele introdus. Daca există deja, utilizatorul va fi redirecționat înapoi la „register.php” unde va fi notificat de această problemă. Altfel, datele vor fi introduse în tabelul „users” (parola va fi criptată și stocată ca un hash md5) și utilizatorul va fi redirecționat către pagina de login.

Logarea se face accesând pagina „login.php”, care este similară cu „register.php”. Avem un formular cu 2 input-uri în care introducem numele, respectiv parola. După ce apasă pe log in, dacă a introdus corect datele, utilizatorul va fi redirecționat la tasklist cu numele și id-ul contului pe care s-a logat în sesiune. Dacă a introdus ceva greșit, acesta va fi trimis înapoi la pagina de login și va fi atenționat de această problemă. Când ajunge pe tasklist, în partea de stânga sus se află un meniu dropdown (numele utilizatorului cu o săgeată în dreapta lui care arată în jos), pe care dacă pune mouse-ul, va arăta un buton de log out, ce îl va trimite la „logout.php” și va distruge sesiunea, după care va trimite utilizatorul înapoi la pagina de log in.

Adăugarea unui task se face prin introducerea a ceva în cutia de input, alegerea unei categorii in care se va adăuga și apăsarea butonului „Add”. Această ultimă acțiune va apela funcția „addTask()”, care va lua valoarea input-ului, valoarea categoriei care este arătată acum și id-ul categoriei selectate. În funcția AJAX se vor trimite la „addtask.php” „task\_string” (care va conține valoarea input-ului) și „category\_id” (care va avea id-ul categoriei selectate). În „addcategory.php” se va prelua „task\_string”, „category\_id” și id-ul utilizatorului preluat din sesiune. Se va da query cu o comandă de inserare SQL și se vor insera string-ul taskului adăugat, id-ul categoriei de care acesta aparține și id-ul utilizatorului căreia ii aparține, împreună cu un id unic și cu statusul task-ului 0 (adică e un task „to do”) în tabela „tasks”. Dacă query-ul a avut loc cu succes, se va stoca într-un array id-ul task-ului care tocmai a fost adăugat și faptul ca nu a fost nici o eroare, iar apoi trimis înapoi la funcția „addTask()”. Funcția va continua prin a verifica dacă a fost vreo eroare. Dacă da, ne atenționează, altfel adaugă în lista „To do” task-ul nou, iar apoi declanșează o schimbare la numele categoriei care este arătată în acest moment. Lista își ia un refresh (cu ajutorul funcției „on change”) și va fi afectată sau neafectată (dacă nu a fost inserat elementul în ea). Operația de adăugare a categoriei este similară.

Editarea unui task sau a unei categorii se fac aproape similar. La apăsarea butonului „Edit”, se va apela funcția „editCategory()” sau „editTask(el)”. Se va lua input-ul și id-ul categoriei sau task-ului. Se va verifica dacă elementul are clasa „editMode”. Dacă nu o are, input-ul ascuns (cu ajutorul căreia edităm task-ul sau categoria) va lua numele categoriei sau task-ului și se va arăta, clasa sa fiind schimbată în „editMode” (ce ii va schimba proprietatea de display în CSS). După ce a fost inserat un nume nou sau editat numele existent în input, apăsarea butonului edit va apela din nou aceeași funcție, care va verifica dacă conține clasa „editMode” sau nu. Dacă o conține, funcția AJAX va trimite noua valoare introdusă și id-ul task-ului sau categoriei în „updatetask.php” sau „editcategory.php”. În aceste fișiere php se vor prelua datele trimise și vor rula o comandă SQL de update, în care vor schimba numele task-urilor cu id-ul primit în tabelul „tasks”. După aceea se va trimite înapoi dacă a fost vreo eroare sau nu și dacă a rulat cu succes, label-ul task-ului sau numele categoriei selectate vor fi modificate în noua valoare. În final, clasa va fi schimbată înapoi în cea pe care o avea înainte.

Ștergerea unui task, a unei categorii sau a mai multor taskuri se fac în mod similar. La apăsarea butonului „Delete”, se va apela funcția respectivă („deleteTask” pentru „Delete”, „deleteCategory” pentru „Delete Category” și „DeleteGeneral” cu id-ul listei pentru care s-a făcut selecția pentru ștergere cu ajutorul funcțiilor „DeleteSelectedToDo” și „DeleteSelectedDone”). Acestea vor lua id-ul categoriei, taskului/taskurilor și le vor trimite cu ajutorul funcției AJAX în fisierul php respectiv fiecărei funcții, unde va fi apelat un query cu comanda SQL de DELETE. Taskul sau taskurile cu id-ul respectiv vor fi șterse, însă dacă o categorie este ștearsă, pe lângă această acțiune se va mai verifica și modifica id-ul categoriei fiecărui task care era asignat la acea categorie în id-ul categoriei „All” (categoria în care apar toate taskurile). Din nou trimitem dacă a fost vreo eroare sau nu la query. Dacă ștergerea s-a făcut cu succes, toate task-urile sau categoriile care au fost șterse în baza de date vor fi șterse și din site. Dacă lista este complet goală, o verificare va da „fade in” la un h3 existent în interiorul listei care informează utilizatorul de acest lucru.

Mutarea taskurilor se face prin butoanele „Move Selected To Done” și „Move Selected Back”. La apăsarea acestora, funcțiile „moveToDone”, respectiv „moveToBack” vor fi apelate, acestea la rândul lor apelând funcția „moveGeneral” cu id-ul listei de la care le mută, id-ul listei la care le mută și statusul pe care îl vor avea taskurile mutate. Funcția „moveGeneral” va verifica mai întâi care elemente din lista categoriei selectate au căsuțele marcate și le va pune într-un array, iar id-ul lor va fi de asemenea pus într-un alt array. Funcția AJAX va trimite id-ul și statusul pe care trebuie să îl aibă taskurile după mutare la „movetasks.php”, unde vor fi preluate și va avea loc un query ce va rula comanda SQL de UPDATE, ce va schimba statusul taskurilor în tabela „tasks” în valoarea nouă. Dacă query-ul a avut loc cu succes, se va trimite faptul că nu a avut loc nici o eroare înapoi la funcția AJAX. În continuare verificăm dacă am primit vreo eroare înapoi, iar dacă nu, înseamnă că totul a fost ok și putem să mutam elementele fără problemă. De asemenea verificăm dacă vreuna din liste este goală pentru a afișa din nou h3-ul care informează utilizatorul.

Schimbarea categoriilor se face prin butoanele „Change Category” care apelează funcția „changeCatToDo” sau „changeCatDone” (depinzând de lista în care facem schimbarea). Aceste două funcții vor apela funcția „changeCatgen” care va lua id-ul cutiei de select din care va trebui să ia datele și id-ul listei a cărei elemente selectate li se vor schimba categoria. Se face un array cu id-urile elementelor selectate, iar apoi se trimit ca și date id-ul categoriei în care le vom schimba din cutia de selecție și id-urile elementelor selectate în „changecategory.php”. În acest fișier va avea loc un query în care se va rula comanda SQL de update pentru tabelul „tasks”, în care va căuta elementele selectate cu ajutorul id-ului și va modifica id-ul categoriei la care erau asignate. Dacă query-ul are loc cu succes, va trimite asta înapoi iar în funcția AJAX în continuare se va verifica acest lucru. Daca totul a fost bine, se va notifica succesul și se va declanșa o schimbare la categorii pentru a da „refresh” la lista în care a avut loc acțiunea de „Change category”.

Tehnologii utilizate în realizarea aplicației

În mare parte pentru realizarea aplicației sunt folosite Javascript (în mare parte JQuery) pentru modificările front-end. PHP a fost folosit pentru modificările bazei de date și pentru afișarea/verificarea informațiilor salvate în baza de date și afișarea diferitor mesaje de eroare la logare/înregistrare. MYSQL a fost folosit pentru baza de date. CSS a fost folosit pentru înfrumusețarea paginilor, pentru a le face cat mai responsive și pentru a putea folosi un font luat de pe fonts.google.com.