Taskuri:

Etapa 0 (0.3)

1. Alegerea temei
2. Descrierea succintă a temei (google docs, 1/2-1 pagina). Va contine următoarele informații:
    a. (0.02) Impartirea informatiilor, serviciilor etc. pe categorii si subcategorii.
    b. (0.03) Identificarea efectiva a paginilor (încercați să fie doar 4-5) și partial a legaturilor dintre ele (ce informatii vor avea pagini separate si care informații vor fi subsecțiuni în aceeași pagină
    c. (0.05) Stabilirea cuvintelor/sintagmelor cheie (o lista cu cuvintele cheie ale site-ului cât și cate o lista pentru fiecare pagină în parte)
    d. (0.2) Căutarea unor site-uri similare (4-5 site-uri) ca tema pentru a observa modul de organizare a informației. Veti observa pentru ficare site cum au impartit informatiile si cum au facut designul. Veti nota pentru fiecare site ideile demne de implementat, dar și neajunsurile acelor site-uri (pentru fiecare site minim 2 lucruri pro si 2 contra). Veți pune linkuri către ele în fișier.

Etapa 1 (punctaj recomandat 0.5)

Creați prima pagină a site-ului (**doar prima pagină**; **fără stilizare** încă, fiindcă veți primi taskuri legate de acest aspect). Puteți pune în această pagină text care va fi mutat în alte pagini, mai târziu, dar **nu faceți încă mai multe pagini fiindcă le vom genera prin Node!** La prezentare vă rog să aveți pentru fiecare task notată linia din program la care l-ați rezolvat ca să nu dureze prezentarea mai mult de 3-4 minute.

1. Creați un folder al proiectului care va cuprinde toate fisierele necesare site-ului vostru. Creați în el un fisier numit index.html. Deschideți acest fișier cu un editor de text care marchează sintaxa. Adăugați în fișier doctype și setați limba documentului în tagul html
2. Adaugati un title corespunzător conținutului textului. Folosiți 4 taguri meta relevante pentru a specifica: charset-ul, autorul, cuvintele cheie, descrierea.
3. Creați un folder (de exemplu numit "resurse") care va conține toate fișierele folosite de site, dar care nu sunt pagini html (de exemplu imagini, fisiere de stilizare etc). In el creati un folder numit **ico**. Adaugati un favicon relevant pentru temă. Folosiți https://realfavicongenerator.net pentru a genera toate dimensiunile necesare de favicon și codul compatibil pentru diversele browsere și sisteme de operare. Pentru favicon transparent, trebuie sa setati si o culoare a tile-ului (de background), care trebuie specificata și în tagul meta: <meta name="msapplication-TileColor" content="...culoarea aleasa de voi...">
4. Împărțiți body-ul în header, main, footer.
5. În **header** faceți un sistem de navigare ca în curs (nav cu listă neordonată de linkuri), cu opțiuni principale (care vor reprezenta paginile site-ului) și secundare (pentru opțiunea "Acasă", adică pagina principală, subopțiunile vor cuprinde linkuri către secțiunile paginii, care vor avea id-uri relevante). Folosiți în header h1 pentru titlul site-ului.
6. Folosiți minim un tag dintre: section, article, aside. Trebuie să existe măcar un caz de taguri de secționare imbricate (secțiune în secțiune). Puneți headingul cu nivelul corespunzător nivelului imbricării. Atenție, nu folosim headinguri decât ca titluri pentru tagurile de secționare. **Observație:** nivelul headingului trebuie să corespundă nivelului de imbricare a secțiunii (de exemplu un tag de secționare aflat direct în body are titlul scris cu h2, dar un tag de sectionare aflat intr-un tag de secționare care la rândul lui se află în body, va avea titlul scris cu h3
7. În cadrul secțiunilor folosiți minim 2 taguri dintre următoare taguri de grupare: p, ol, ul, blockquote, dl
8. Adăugați în pagină o imagine cu descriere, folosind figure și figcaption. Pe ecran mic (mobil) trebuie să se încarce o variantă mai redusă în dimensiune (bytes) a imaginii, pe tabletă o variantă medie, iar pe ecran mare varianta cea mai mare a imaginii. Folosiți un editor grafic pentru cropping și redimensionare pentru a obține cele 3 variante de imagini.
9. Textul trebuie să conțină toate cuvintele cheie identificate pentru pagina curentă. Puteți găsi mai multe sintagme cheie pe care le puteți folosi, cu **https://www.wordtracker.com/** sau https://app.neilpatel.com/en/ubersuggest/keyword_ideas

    Acestea trebuie să apară de mai multe ori în pagină, în taguri relevante.

10. În cadrul textului îndepliniți **3** dintre cerințele de mai jos, **la alegere**:
    a. marcați cuvintele și sintagmele cheie cu ajutorul tagului b
    b. marcați textul idiomatic (termeni științifici, în altă limbă, tehnici, de jargon, etc) cu tagul i
    c. marcați textul de atenționare cu strong
    d. marcați textul accentuat cu em
    e. marcați textul șters (corectat sau care nu mai e relevant) cu tagul s și textul inserat în loc cu tagul ins
    f. marcați o abreviere cu abbr și cu atributul title specificați sintagma abreviată
    g. marcați un termen definit cu dfn
    h. marcați un citat cu tagul q
11. Creați următoarele linkuri speciale:

a. un link extern (va fi in continutul paginii, nu in meniu, va face referire la alt site si se va deschide in fereastră nouă)
b. un link în footer către începutul paginii,
c. minim două linkuri care se deschid într-un iframe (se poate face ca în exemplul de curs, linkuri care deschid videoclipuri relevante de pe youtube în iframe). Atentie nu e vorba de src-ul iframe-ului ce de taguri <a> care la click se deschid in iframe. Iframe-ul va contine in mod default una dintre resursele specificate in linkuri
d. Un link de tip download

12. Creați în pagină mai multe zone de details și summary. Pot fi întrebări frecvente, pot fi niște oferte pentru care afișăm titlul și utilizatorul le deschide pe cele care îl interesează, pot fi secțiuni explicative etc.
13. În footer se vor adăuga cu ajutorul tagului address informații de contact:
    a. telefon fictiv, marcat cu tagul <a> si URI Scheme-ul corespunzător
    b. adresă fictivă care la click deschide o locatie pe Google Maps (locatia in mod normal ar corespunde cu adresa dar voi veti pune drept locatie in maps, Facultatea de matematica si informatica)
    c. e-mail fictiv, marcat cu tagul <a> si URI Scheme-ul corespunzător în href
    d. Link care deschide o aplicatie de comunicare precum skype sau whatsapp pentru chat
14. În footer se va adăuga informație de copyright, folosind tagul small, simbolul specific de copyright cu codul html necesar (forma &cod;) și data creării paginii scrisă în limba română și pusă în tagul time cu **atributul datetime** corespunzător.
15. Pagina trebuie sa fie valida din punct de vedere sintactic. Deci verificati cu validatorul html. Validatorul va fi pregătit într-un tab, la prezentare, și pagina se va valida pe loc.

Se poate da bonus pentru o temă bine făcută sau pentru folosirea mai multor taguri decât minimul specificat (dar studentul trebuie să anunțe la prezentare că le-a folosit)

Evitați pentru moment să adăugați alte taguri fiindcă vor apărea în taskurile următoare.

Etapa 2 (0.5p)
Atentie - unele cerinte au enunt diferit pentru fiecare student (si sunt marcate printr-un link). Trebuie sa va inregistrati pe site pentru a le vedea.
Daca stilizarea dintr-o cerinta nu va place, puteti sa imi cereti o alta varianta (imi scrieti pe chat). Culorile din imaginile si videoclipurile date ca exemplu nu trebuie respectate (folositi culorile din schema cromatică aleasă de voi).
(0.025) Task schema cromatica: (cerință individuală)
(0.15) Task layout: (cerință individuală)
(0.2) Task meniu: (cerință individuală)
(0.05) Task taburi iframe: (cerință individuală)
(0.05) Link top: (cerință individuală)
(0.025) Task iconuri și font extern. Folosiți in prima pagină a site-ului un font extern: Google API. Folosiți în pagină, într-un loc relevant un icon static și unul animat (separat de eventuale alte taskuri care cer așa ceva) din colecția Font Awesome

Etapa 3 (punctaj recomandat 0.5)
(0.25)**Trecerea site-ului pe node** și crearea de fișiere EJS conform cerințelor:

1. Se va crea un obiect server express care va asculta pe portul 8080. (sau alt port daca aveti deja folosit 8080)
2. Se va folosi EJS pentru generarea (randarea) paginilor. Se va face un folder numit views în rădăcina proiectului. În el veți face un folder numit pagini (care conține paginile întregi) și altul numit fragmente (care conține părți de pagini (bucățele de cod html) ce pot fi refolosite pe mai multe pagini).
3. Din index (care va fi redenumit index.ejs) se vor decupa headerul și footerul și se vor pune în ejs-uri separate. De asemenea se va decupa partea de head care conține codul care nu se schimbă în funcție de pagină (de exemplu, tagul meta cu encodingul sau autorul, includerea faviconului, fișierelor css generale (nu specifice paginii) a scripturilor generale etc). Se va folosi funcția include() pentru a include toate aceste fragmente în pagini
4. Se va realiza (dacă nu l-ați făcut deja) un folder special cu toate resursele site-ului (în stilul exemplului de la curs în care am pus toate fișierele statice, precum imagini, fișiere de stil, videoclipuri etc în folderul "resurse"). Numele folderului îl decideți voi, însă va trebui să fie structurat, de asemenea, în subfoldere în funcție de tipul și modul de utilizarea al fișierelor. Se va defini în program acest folder ca fiind static
5. Se vor schimba căile fișierelor-resursă folosite în pagini, astfel încât să nu mai fie relative ci stil cerere către server (de exemplu, /resurse/stiluri/ceva.css în loc de, de exemplu, ../resurse/stiluri/ceva.css)
6. Prima pagină (index) trebuie să se poată accesa atât cu localhost:8080 cât și cu localhost:8080/index, localhost:8080/home.

7. Veți declara un app.get() general care tratează orice cerere de forma /pagina randând fișierul pagina.ejs. Dacă acesta nu există, se va randa o pagină specială de eroare 404. Se va crea o pagină specială pentru eroarea 404. Pagina va avea atât ca titlu cât și heading textul "Eroare 404". Pagina pentru 404 trebuie să conțină headerul (cu tot cu meniu) și footerul. Pagina va avea un text care să se potrivească cu tema site-ului și o imagine care să simbolizeze căutarea fără succes.Se va seta codul de eroare 404 al răspunsului folosind funcția status().
8. Veți mai face încă o pagină (cu puțin text sau imagini, ca să aibă conținut), de exemplu o pagină cu descrierea site-ului sau istoricul său, al firmei virtuale pentru care este făcut etc. Această pagină trebuie să poată fi accesată prin meniu (linkul să fie corect și să transmită o cerere de tip get). Nu faceti încă pagina de produse, findcă pe acelea le preluăm din baza de date. Nici paginile de înregistrare sau login, fiindca le tratăm separat.
9. În zona din layout de date despre utilizator vom afișa ip-ul utilizatorului (prin program). Deocamdată, site-ul fiind local, veți vedea mereu ip-ul de localhost (adică ::1). Ip-ul real se va vedea când adăugați site-ul pe Heroku.
10. La cererea oricărui fișier cu extensia ejs se va transmite o eroare de tip 403 Forbidden. Pagina de 403 va avea format similar cu cea de 404, dar textul și imaginea schimbate corespunzător.

(0.05) Task tabel: (cerință individuală)
(0.05) Se va adăuga in pagina un **video** cu conținut relevant pentru site, conform (cerință individuală)
(0.15) Să se realizeze un stil pentru printare conform (cerință individuală)
(0.15) Task **optional (e pe lângă punctajul recomandat)** hr: (cerință individuală)

Etapa 4 (punctaj recomandat 0.5)

1. (0.15) Să se realizeze un view (template) de eroare generala care va fi folosit pentru toate erorile din program (inclusiv 403 și 404). Va exista un JSON cu descrierile erorilor (identificator, titlu, explicatie, imagine). În program veți trimite pe baza JSON-ului ce să se afișeze în view-ul de eroare (de exemplu pentru eroare 404, veți transmite informații de la identificatorul "404" din JSON). Cel mai elegant ar fi să rezolvați cu ajutorul unei funcții care primește obiectul de tip raspuns și identificatorul erorii și randează pagina.
2. (0.35) Galeria statica (cerință individuală)

Etapa 5 (punctaj recomandat 0.5)

Galeria animată (cerință individuală)

Etapa 6 - afișarea produselor (punctaj recomandat 1.2) - se porneste de la codul dat ca exemplu in curs.
1 afisare + sortare/filtrare/calculare (cerință individuală)
0.2 adăugare pe Heroku

bonus (0.1) daca se genereaza inputurile pe baza informatiilor din tabel (precum select-ul)

Etapa 7 (punctaj recomandat: 3.5)
Atentie! Nu implementați de la zero această etapă. Porniți de la arhiva-exemplu de pe drive și doar preluați codul in proiectul vostru și completați/modificați conform cerințelor. Etapa 6 va avea codul 80-85% scris deja de mine in timpul cursurilor si laboratoarelor. Voi aveti doar de modificat/completat!

(0.2p) light/dark theme cu variabile CSS (optional in SASS). Pentru moment va exista un buton în pagină care va face schimbarea. Butonul va fi reprezentat printr-o imagine cu soare (pt light) vs lună (pt dark) - poate fi un icon din fontawesome. Tema aleasă se va memora in localStorage si se va pastra tema la urmatoarea intrare pe pagina si pe restul paginilor site-ului.
(0.3p) Task bootstrap (cerință individuală) - daca ati folosit bootstrap in alta parte in proiect si vreti sa faceti cerinta legata de acel caz, putem discuta sa o modificam sa completati la ce aveti deja facut cu eventualele obiective suplimentare (daca nu sunt deja atinse).
(0.3p) Task banner (cerință individuală)
(2.2p) Task utilizatori+cos virtual+evenimente (cerință individuală) Punctajul pentru fiecare subpunct este: 1(0.05) + 2.(0.05) + 3(0.05) + 4(0.2) + 5(0.1) + 6(0.15) + 7(0.05) + 8(0.2) + 9(0.15) + 10(0.1) + 11(0.1) + 12(0.05) + 13(0.25) +14 (0.05) + 15(0.1) + 16 (0.1) + 17 (0.05) + 18(0.05) + 19(0.2) + 20(0.05)+ 21 (0.05) + 22 (0.05)

(0.5p) (**evaluarea finală a site-ului** - nota se da pe: aspect, accesibilitate, viteza de încărcare, layout (responsive).

Bonusuri: (punctajele acestora sunt pe lângă cele 3.5p enunțate - deci puteți face cerințe la alegere -- dacă nu vă place ceva din cerințele recomandate, puteți înlocui cu o cerință bonus-- sau puteti propune voi o functionalitate, dar intai o discutam si apoi o implementati, si-i decidem si punctajul cand o discutam). Dacă luați mai multe puncte pe proiect, puteți acoperi punctajul lipsa din alte parti ale notei finale (activitate sau examen).
1. (0.15) **Criptarea parolelor sa se faca pentru fiecare utilizator cu un alt *salt string*** (parola de criptare). Salt-ul se va genera aleator si se va memora in tabel pentru fiecare utilizator. La login se preia salt-ul din baza de date si se cripteaza cu ajutorul lui parola primita din

input. Se compara parola criptata astfel cu cea din tabel.
2. (0.15) **Utilizatorul să poată alege dintre 3 sau mai multe teme** (nu doar light/dark). Ca și în cazul light/dark tema aleasă se memorează în localStorage. Modul de alegere a temei se poate realiza printr-un select simplu sau radio buttons.
3. (0.2) Dacă utilizatorul alege tema (light/dark sau o alta) în timp ce este logat, **tema se va memora pe server** pentru el (de exemplu în tabel) și de fiecare dată când se loghează se încarcă tema preferată (se schimbă automat la login, fără ca utilizatorul să dea click pe butonul de schimbare a temei.
4. (0.3) **Stocuri limitate pentru produse.** Va exista si o coloană în care e trecut stocul pentru fiecare produs. Adminul are dreptul de a modfica stocul (o pagină specială accesibilă doar adminului în care poate actualiza stocul). Utilizatorul nu poate cumpăra mai multe produse decât sunt în stoc. Se va afișa pentru fiecare produs câte mai sunt disponibile.
5. (0.2) Implementați o **metodă de recuperare/resetare a parolei** la cererea utilizatorului.
6. (0.25) Implementati o **metodă automată de ștergere a contului**, la cererea utilizatorului. Utilizatorul trebuie sa confirme apoi dand un click pe un link primit pe mail.
7. (0.1) Implementați în pagina de profil o metodă de schimbare a parolei.
8. (0.2) **Dacă cineva greșește parola de k ori** într-un timp t (de exemplu, k=5 si t=10min) atunci **login-ul să fie blocat pentru un timp T**(de exemplu T=1ora) iar utilizatorul să primească un mail de avertizare.
9. (0.2) **Se va implementa opțiunea de "site în mentenanță".** Se va crea un fișier JSON cu opțiuni pentru server. Se va crea un middleware cu app.use() pentru orice cerere (care va apela next() pentru a transmite mai departe cererea după ce i-a aplicat opțiunile). Dacă în JSON-ul de opțiuni, opțiunea de site în mentenanță este true, atunci orice cerere primeste ca raspuns o pagina speciala care anunta acest lucru, altfel e forwardata cu next().
10. (0.1) Afișați pe prima pagină pentru utilizator **cât timp a trecut de la ultima lui logare pe site**
11.(0.2) Implementați opțiunea de a **rămâne logat pe site automat** (de exemplu în funcție de ip).
12.(0.2) **Marcarea în meniu a paginii curente** (de exemplu cu o altă culoare dacă utilizatorul se afla chiar la pagina indicată de opțiunea din meniu).
13.(0.15) Când utilizatorul face click pe un link care duce spre o secțiune din aceeași pagină (cu href="#id") **scrollul** către acea secțiune **să se facă animat, treptat**, nu brusc, cum e default.
14.(0.15) **Dacă utilizatorul nu a mai intrat de multă vreme** (alegeți voi intervalul de timp) pe site sau nu a mai cumpărat de multă vreme ceva, **să primească un mail promoțional** care să-i amintească de site. Mailul trebuie să cuprindă minim o imagine.
15. (0.15-0.55 in functie de cat de complex e facut) **Sa existe mai mult de 2 roluri posibile pentru utilizatori** (de exemplu, admin, comun si moderator). Se da bonus suplimentar daca fiecarui utilizator i se aloca un set de drepturi si verificarile de acces se fac pe baza drepturilor si nu a rolului. Se vor defini niste actiuni speciale pe care doar anumite roluri le pot face pentru a arata diferenta intre roluri (de exemplu moderatorul poate sterge mesaje de pe pagina de contact, dar nu are acces la pagina cu utilizatorii).. Adminul are insa toate drepturile posibile.

Etapa 8 (optionala; punctaj recomandat: ~~2p~~ 1p)
(0.5) introduceti o pagina de comentarii (precum cea din exemplu, de la pagina "contact"). Culoarea de text a comentariului va fi in functie de rolul utilizatorului (de exemplu admin scrie cu rosu, iar utilizatorii obisnuiti cu negru)
(0.7) cerinta custom chat (cerință individuală)