In this document we will have the division of the project into stages with details about the tasks (some tasks are general - the same for all students - and others have individual requirements and require registration on the subject website)

Useful links for the project:
https://pixabay.com/ (free images)
https://www.videvo.net/ (free videos - those marked with "free")

Pocket:

Stage 0 (0.3)

1. Choosing the theme
2. Brief description of the theme (google docs, 1/2-1 page). It will contain the following information:
   a. (0.02) Sharing of information, services, etc. by categories and subcategories.
   b. (0.03) Effective identification of the pages (try to be only 4-5) and partially of the links between them (which information will have separate pages and which information will be subsections on the same page
   c. (0.05) Establishing keywords/phrases (a list of website keywords and a list for each page)
   d. (0.2) Searching similar sites (4-5 sites) as the theme to observe how the information is organized. You will notice for the site how they shared the information and how they did the design. You will note for each site the ideas worthy of implementation, but also the shortcomings of those sites (for each site at least 2 pros and 2 cons). You will put links to them in the file.

Stage 1 (recommended score 0.5)
Create the front page of the site (**only the first page**; **without styling** yet, because you will receive tasks related to this aspect). You can put text on this page that will be moved to other pages later, but**don't make more pages yet because we'll generate them through Node!** During the presentation, please note for each task the line in the program where you solved it so that the presentation does not last more than 3-4 minutes.

1. Create a project folder that will contain all the files needed for your website. Create a file in it called index.html. Open this file with a syntax highlighting text editor. Add to file doctype and set document language in html tag
2. Add a title corresponding to the content of the text. Use 4 relevant meta tags to specify: charset, author, keywords, description.
3. Create a folder (for example called "resources") that will contain all the files used by the site, but which are not html pages (eg images, styling files, etc.). In it you create a folder called**ico**. Add a relevant theme favicon. Usehttps://realfavicongenerator.net to generate all required favicon sizes and compatible code for various browsers and operating systems. For a transparent favicon, you must also set a color of the tile (background), which must also be specified in the meta tag: <meta name="msapplication-TileColor" content="...the color of your choice..." >
4. Split the body into header, main, footer.
5. In the**header** make a navigation system as in progress (nav with unordered list of links), with main options (which will represent the pages of the site) and secondary options (for the "Home" option, i.e. the main page, the sub-options will contain links to the sections of the page, which will have relevant ids). Use in h1 header for site title.
6. Use at least one tag from: section, article, aside. There must be at least one case of nested section tags (section within section). Set the heading with the level corresponding to the nesting level. Note, we only use headings as headings for section tags.**Observation:**the heading level must correspond to the nesting level of the section (for example a section tag located directly in the body has the title written with h2, but a section tag located within a section tag that is in turn located in the body will have title written with h3
7. Within the sections, use at least 2 of the following grouping tags: p, ol, ul, blockquote, dl
8. Add a description image to the page, using figures and figcaption. On a small (mobile) screen, a version smaller in size (bytes) of the image must be loaded, on a tablet a medium version, and on a large screen the largest version of the image. Use a graphics editor for cropping and resizing to get the 3 image variants.
9. The text must contain all the keywords identified for the current page. You can find several key phrases that you can use, with**https://www.wordtracker.com/** orhttps://app.neilpatel.com/en/ubersuggest/keyword_ideas

   They must appear several times on the page, in relevant tags.

10. Within the text fulfill**3** of the requirements below,**at your choice**:
    a. tag key words and phrases using the b tag
    b. tag idiomatic text (scientific terms, in another language, techniques, jargon, etc) with the i tag
    c. mark the warning text with strong
    d. mark the highlighted text with em

    e. mark deleted text (corrected or no longer relevant) with the s tag and inserted text with the ins tag
    f. mark an abbreviation with abbr and with the title attribute specify the abbreviated phrase
    g. mark a defined term with dfn
    h. mark a quote with the q tag
11. Create the following special links:
    a. an external link (it will be in the content of the page, not in the menu, it will refer to another site and will open in a new window)
    b. a footer link to the top of the page,
    c. at least two links that open in an iframe (can be done as in the course example, links that open relevant youtube videos in an iframe). Attention, it is not about the src of the iframe, but about the <a> tags that open in the iframe when clicked. The iframe will by default contain one of the resources specified in the links
    d. A download link
12. Create several details and summary areas on the page. It can be frequently asked questions, it can be some offers for which we display the title and the user opens the ones that interest him, it can be explanatory sections, etc.
13. In the footer, contact information will be added using the address tag:
    a. fictitious phone, marked with the <a> tag and the corresponding URI Scheme
    b. fictitious address which, when clicked, opens a location on Google Maps (the location would normally correspond to the address, but you will put the Faculty of Mathematics and Informatics as the location in maps)
    c. fictitious email, marked with the <a> tag and the corresponding URI Scheme in the href
    d. Link that opens a communication application such as skype or whatsapp for chat
14. In the footer, copyright information will be added, using the small tag, the specific copyright symbol with the necessary html code (&cod; format) and the page creation date written in Romanian and placed in the time tag with**datetime attribute**suitable.
15. The page must be syntactically valid. So check with validatorul html. The validator will be prepared in a tab, upon presentation, and the page will be validated on the spot.

A bonus can be given for a well-done assignment or for using more tags than the specified minimum (but the student must announce at the presentation that they have been used)

Avoid adding other tags for now because they will appear in the next tasks.

Stage 2 (0.5p)
Attention - some requirements have a different statement for each student (and are marked by a link). You have to register on the website to see them.
If you don't like the styling from a request, you can ask me for another option (write to me on chat). The colors in the images and videos given as an example must not be respected (use the colors from the color scheme chosen by you).
(0.025) Task color scheme: (individual requirement)
(0.15) Task layout: (individual requirement)
(0.2) Task menu: (individual requirement)
(0.05) Task sprinkle iframe: (individual requirement)
(0.05) Link top: (individual requirement)
(0.025) Task icons and external font. Use an external font on the first page of the site: Google API. Use on the page, in a relevant place, a static icon and an animated one (separate from any other tasks that require this) from the Font Awesome collection

Stage 3 (recommended score 0.5)
(0.25)**Moving the site to node** and creating EJS files as required:
1. An express server object will be created that will listen on port 8080. (or another port if you have already used 8080)
2. EJS will be used to generate (render) the pages. A folder called views will be made in the root of the project. In it you will make a folder called pages (which contains the whole pages) and another called fragments (which contains parts of pages (pieces of html code) that can be reused on several pages).
3. From the index (which will be renamed index.ejs) the header and footer will be cut and placed in separate ejs. Also, the part of the head that contains the code that does not change according to the page will be cut (for example, the meta tag with the encoding or the author, the inclusion of the favicon, the general css files (not specific to the page) of the general scripts, etc.). The include() function will be used to include all these fragments in the pages
4. It will create (if you haven't already) a special folder with all the site's resources (in the style of the example from the course where I put all the static files, such as images, style files, videos, etc. in the "resources" folder). The name of the folder is up to

you, but it will also need to be structured into subfolders based on the type and usage of the files. This folder will be defined as static in the program

5. Will change resource file paths used in pages so they are no longer relative but server request style (eg /resources/styles/something.css instead of eg ../resources/styles/something .css)
6. The first page (index) must be accessible with both localhost:8080 and localhost:8080/index, localhost:8080/home.
7. You'll declare a general app.get() that handles any request of the form /page by rendering the page.ejs file. If it does not exist, a special 404 error page will be rendered. A special page for the 404 error will be created. The page will have both the title and heading text "Error 404". The page for 404 must contain the header (including the menu) and the footer. The page will have text to match the theme of the site and an image to symbolize the unsuccessful search. The 404 error code of the response will be set using the status() function.
8. You will make one more page (with little text or images, so that it has content), for example a page with the description of the site or its history, the virtual company it is made for, etc. This page must be menu accessible (the link must be correct and send a get request). Do not make the product page yet, because we are taking those from the database. Nor the registration or login pages, because we treat them separately.
9. In the user data layout area we will display the user's ip (by program). For now, since the site is local, you will always see the ip of localhost (ie ::1). The real IP will be seen when you add the site to Heroku.
10. Requesting any file with the ejs extension will return a 403 Forbidden error. The 403 page will have a similar format to the 404, but the text and image changed accordingly.

(0.05) Task table: (individual requirement)
(0.05) It will be added to page one**video** with content relevant to the site, according to (individual requirement)
(0.15) To make a style for printing according to (individual requirement)
(0.15) Task **optional (it's in addition to the recommended score)** hr: (individual requirement)

Stage 4 (recommended score 0.5)

1. (0.15) Create a general error view (template) that will be used for all errors in the program (including 403 and 404). There will be a JSON with the descriptions of the errors (identifier, title, explanation, image). In the program you will send based on the JSON what to display in the error view (for example for error 404, you will send information from the "404" identifier in the JSON). The most elegant solution would be to use a function that receives the response object and the error identifier and renders the page.
2. (0.35) Static gallery (individual requirement)

Stage 5 (recommended score 0.5)
Animated gallery (individual requirement)

Stage 6 - product display (recommended score 1.2) - it starts from the code given as an example in progress.
1 display + sort/filter/calculate (individual requirement)
0.2 addition on Heroku

bonus (0.1) if the inputs are generated based on the information in the table (such as the select)

Stage 7 (recommended score: 3.5)

(0.2p) light/dark theme with CSS variables (optionally in SASS). For now there will be a button on the page that will make the change. The button will be represented by an image of sun (for light) vs moon (for dark) - it can be an icon from fontawesome. The chosen theme will be saved in localStorage and the theme will be kept at the next entry on the page and on the rest of the website pages.
(0.3p) Task bootstrap (individual requirement) - if you have used bootstrap elsewhere in the project and you want to make a request related to that case, we can discuss to modify it to complement what you have already done with possible additional objectives (if they are not already achieved).
(0.3p) Task banner (individual requirement)
(2.2p) Task users+virtual basket+events (individual requirement) The score for each subpoint is: 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) (**final evaluation of the site**- the grade is given on: appearance, accessibility, loading speed, layout (responsive).

Bonuses: (their scores are in addition to the stated 3.5p - so you can make requirements of your choice -- if you don't like something from the recommended requirements, you can replace it with a bonus requirement-- or you can propose a functionality, but first we discuss it and then you implement it, and we decide its score when we discuss it). If you get more

points on the project, you can cover the missing points in other parts of the final grade (activity or exam).
1. (0.15) **Encryption of passwords should be done for each user with a different one*salt string*** (encryption password). The jump will be randomly generated and saved in the table for each user. When logging in, the jump from the database is taken and the password received from the input is encrypted with it. The password encrypted in this way is compared with the one in the table.
2. (0.15) **The user can choose from 3 or more themes** (not just light/dark). As with light/dark, the chosen theme is stored in localStorage. The way to choose the theme can be done by a simple select or radio buttons.
3. (0.2) If the user chooses the theme (light/dark or another) while logged in, **the theme will be saved on the server**for him (for example in the table) and every time he logs in, the preferred theme is loaded (it changes automatically upon login, without the user clicking on the theme change button.
4. (0.3) **Limited product stocks.** There will also be a column in which the stock is entered for each product. The admin has the right to modify the stock (a special page accessible only to the admin where he can update the stock). The user cannot buy more products than are in stock. It will show for each product how many are still available.
5. (0.2) Implement a**password recovery/reset method** at the request of the user.
6. (0.25) Implemented o**automatic account deletion method**, at the request of the user. The user must then confirm by clicking on a link received by email.
7. (0.1) Implement a password change method in the profile page.
8. (0.2) **If someone gets the password wrong k times** in a time t (for example, k=5 and t=10min) then **login to be blocked for a time T**(for example T=1 hour) and the user to receive a warning email.
9. (0.2) **The "site under maintenance" option will be implemented.** A JSON file will be created with options for the server. A middleware will be created with app.use() for any request (which will call next() to forward the request after applying its options). If in the JSON of options, the site under maintenance option is true, then any request receives as a response a special page announcing this, otherwise it is forwarded with next().
10. (0.1) Show on first page for user**how much time has passed since his last login to the site**
11.(0.2) Implement the option to a**stay logged in to the site automatically**(eg by ip).
12.(0.2) **Bookmark the current page in the menu** (for example with a different color if the user was right on the page indicated by the menu option).
13.(0.15) When the user clicks on a link that leads to a section of the same page (with href="#id")**scrollul** to that section **to become animated, gradually**, not suddenly, as is the default.
14.(0.15) **If the user has not logged in for a long time** (you choose the time interval) on the website or has not bought anything for a long time,**to receive a promotional email**to remind him of the site. The email must contain at least one image.
15. (0.15-0.55 depending on how complex it is made)**There should be more than 2 possible roles for users** (for example, admin, common and moderator). An additional bonus is given if each user is assigned a set of rights and access checks are made based on rights and not on role. Some special actions will be defined that only certain roles can do to show the difference between the roles (for example, the moderator can delete messages from the contact page, but does not have access to the user page).. However, the admin has all possible rights .

Stage 8 (optional; recommended score:~~2p~~ 1p)
(0.5) insert a comments page (like the one in the example, from the "contact" page). The text color of the comment will depend on the user's role (for example admin writes in red, and regular users in black)
(0.7) custom chat requirement (individual requirement)