Home page/index page/start page (eg., page user should open first):

http://localhost:4000/index.html
index.js – starts a server.

Using the site - information:

1. Registration and Login. The registration and login form is available in the top navigation bar from any site page. Since it is a learning project, the database was not used for this purpose, so newly added users will only be available as part of the current server session.

2. Site Navigation. The main sections are:
   - main page http://localhost:4000/index.html, from which all functions necessary for ordering products are available;
   - product details page, accessible through a link from the product cards on the main page. Contains a more detailed description of the products and a list of allergens, all the functions needed to order a product are available;
   - shopping cart, accessible after logging in from any page of the website.

3. User functionality that requires primary modification as a part of validation[1] (in order to bring the project to the state of the commercial site):
   - changing the content on the main page to dynamic (to facilitate marketing and administrative tasks);
   - adding the ability to remove items from the shopping cart;
   - adding the ability to Logout;

4. Server functionality which requires primary modification to bring the learning project to the condition of the commercial site:
   - using the database for authorization and registration of users, saving other data needed to complete the purchase business process;
   - using a more advanced user id generator;
   - using the database to save confirmed orders;
   - choosing the most efficient way to ensure that the contents of shopping carts can be stored by a large number of users at the same time (before order confirmation);
   - clearing the Session storage in the case of a Logout user.

[1] Validation is considered as bringing the software product to a state of compliance with the exact needs of the customer.

Implementation details

ITEM 1	Reference
Allow the customer to enter their login details:	See index.html (the modal window called by the menu item in the navigation bar) and the authenticateUser() function in the auth.js module.
Login details validated (via a login screen) before receiving a summary of the order:	Implemented using HTML5 and JavaScript.
Username set to "user"	user
Password set to "pass"	pass
Brief description of implementation details:	Implemented a simple login and password check based on the lecture material. The file index.js contains a route /login processing POST request. Sending authorization request is available from any page of the site. The credentials are validated through the authenticateUser() function in the auth.js module. This function checks the username and password against those contained in the "users" array. Authorization confirmation or rejection is implemented via rendering to greetings.ejs, login_failed.ejs or cart&greetings.ejs (in case some products have already been selected). Also, when any page is loaded, a script triggers once and assigns an ID to the current user, which is stored in the Session storage and later serves to identify the order.

ITEM 2	Reference
Perform form validation through JavaScript or HTML to ensure that text fields are not empty, and a valid email address is entered:	See the modal window invoked through the menu item in the navigation bar on the index.html page or any other page of the site.
Brief description of implementation details:	Implemented with HTML. Added attribute "required" to data entry fields (to ensure that text fields are not empty). The email field is typed as email, which performs validation through HTML to ensure that entered email address is valid.

ITEM 3	Reference
Include a slideshow or carousel which displays a different image each time the page is loaded;	See. index.html and greetings.ejs, script.js
Brief description of implementation details:	Implemented using the Bootstrap component and adding JavaScript. To change the first image in the carousel every time the page reloads made by dynamically adding the active class to the corresponding carousel tag (uses DOM request). The class name is chosen using Math.random().

ITEM 4	Reference
Allow the user to 'purchase' items from the site:	See product cards at index.html, the "buy" item in the "accordion" element of the produst_details.ejs template, as well as cart.ejs and cart.js.
Brief description of implementation details:	The user can choose a product and its quantity through the product card on the index.html page or in the "buy" item in the "accordion" element in the template produst_details.ejs. After pressing the "Add to cart" button a GET request is sent which is processed by the "/addItem" route. It refers to the cart.js module, within which the current state of the customer's cart is saved. When going to the cart (the "/cart" route), the cart.ejs template is rendered, inside of which using the cycle and referring to the variable "items" of the cart.js module displays a list of selected items and the cost of the order.

ITEM 5	Reference
Use an object or an array in JavaScript;	See cart.js (items) and auth.js (users, authorizedUsers).
Brief description of implementation details:	This project objects and arrays in custom modules uses as a simplified alternative to databases.

ITEM 6	Reference
Use at least one custom module in node;	See cart.js and auth.js.
Brief description of implementation details:	Implemented two custom modules: cart.js - stores the list of items selected in the current session. It contains the addItem() function, which adds data about items to the "items" object, accordingly (depending on the current state of the "cart"). This function automatically calculates the number of ordered items and their total price. In the following, the information from the items variable will be used to output the order to the user in the cart.ejs. auth.js. - stores data about the registered users and the users authorized in the current session. It contains the createUser() function which adds the registration data of a new user to the "users" variable; and the authenticateUser() function which implements a simple username and password check against the "users" variable and, if successful, adds this user (his name and the ID assigned in the session) to the list of authorized users (the "authorizedUsers" variable).

ITEM 7	Reference
Include capability for handling post and get requests;	See index.js and related HTML forms.
Brief description of implementation details:	POST and GET requests are performed in the following cases: - user authorization (/login route), POST request is sent from form (id="login-form") from any page of the site; - user registration (/register route), POST request is sent from the form (id="login-form") from any page of the site - adding items to the cart (the route /addItem), GET request is sent from the form (id="add-item") from page index.html and template producst_details.ejs; - go to cart (/cart route), GET request is sent by clicking on "Cart" item in "Navbar" from any page of the site; - transition to the product details page (route /get_more_details), GET request is sent by clicking on the link "More details" in the product card on the index.html page.

ITEM 8	Reference
Include both static and dynamic content;	See index.html and the contents of the views directory.
Brief description of implementation details:	The following pages are composed most of the static content: index.html, greetings.ejs, login_required.ejs, registration_success.ejs, login_failed.ejs. For index.html such a method was chosen in training purposes, in terms of a really functioning commercial site, it can not be considered the best implementation, as there may be a number of difficulties with updating the content (in the product cards). The dynamic content contains: cart.ejs, cart&greetings.ejs (which generate and display the details of the selected products and their total cost under the heading Your order: .), producst_details.ejs (output image, detailed description and other information about the product in the "Accordion" component), registration_failed.ejs (variable <%= credential %>). Also on the pages are dynamically changing some of the properties of the tags. So, added scripts that change properties of some form tags (adding the unique ID to the fields with the property id="userId"), this is done to further identify the user and his order. One more script changes link in id="cartRef" tag, thus adding data for HTTP request to server.

ITEM 9	Reference
Include the use of templates in Node;	See the contents of the views directory and index.js.
Brief description of implementation details:	EJS (Embedded JavaScript) was used as the view engine. Templates are rendered in the following cases: - processing request for login (route /login); - processing request for user registration (route /register); - processing the request to go to cart (route /cart); - processing request to go to the product details page (route /get_more_details).

ITEM 10	Reference
Include error messages to provide feedback to user sin case of issues or errors;	See route /login and login_failed.ejs template; route /cart and login_required.ejs template, registration_failed.ejs;
Brief description of implementation details:	In the route /login case, if auth.authenticateUser() returns False, alternative code is executed which renders the login_failed.ejs template and notifies the browser that authorization failed; also, depending on the result returned by the createUser() function, the registration_failed.ejs is rendered with the text of the corresponding registration problem. In the case of route /cart, if the user attempts to access the cart before authorization, the login_required.ejs. template is rendered, displaying a notification in the browser stating that login is required.

ITEM 11	Reference
Connect to a database that contains relevant site information (eg., product info, prices) using NODE (your database name should be your ATU ID);	See the code under the comment "// Connect to database g00425727" in the index.js file.
Brief description of implementation details:	The database 'g00425727' contains data about products. In index.js. connected module mysql, which connects the database and runs queries required for the operation of the site. Based on the lecture material.

ITEM 12	Reference
Use Bootstrap version 5 via CDN	See index.html or any of the contents of the "views" directory, under the code comment "Connecting the Bootstrap CSS master file via CDN".
Brief description of implementation details:	To connect Bootstrap via CDN (i.e. without downloading the library to the server), the code from the official Bootstrap website https://getbootstrap.com/docs/5.3/getting-started/introduction/#cdn-links was added into the <head> section of the HTML documents.

Additional information: as part of this training project, most of the functionality is implemented in a simplified form. As a result of the project the following areas were identified for further study and development: implementation of user authorization and registration through access to the database, rather than variables within the module; more detailed study of ways to use dynamic and static content in order to avoid repeating the same code on different pages (in the current case - footer and header, navbar, correct script response); study more rational ways to implement the possibility of simultaneous ordering items by several users, further validation of the application. etc.