Lappeenrannan teknillinen yliopisto

School of Technology

Software Development Skills Front-End, Online course

Małysa Maria, 003005073

LEARNING DIARY

Software Development Skills: Front-End 2024-25 MODULE

<https://github.com/koszmary/Front-end_2025>

<https://koszmary.github.io/Front-end_2025/PROJECT/>

**LEARNING DIARY**

**Part I – videos**

**10.02.2025**

Video: 1 – HTML, part I

Content:

Definition and history of HTML as markup language, versions of HTML.

What a tag is, their attributes and <>.

Building HTML page.

Common elements.

Tags & their meaning tags.

Element tree.

**11.02.2025**

Video: 2 – HTML, part II

Content:

Importance of using tags in order and using paragraphs.

More about attributes and their classes, links (absolute or relative).

Building a page structure.

Creating custom formatting.

HTML forms.

How to form body.

Difference between Id and name.

Other attributes.

**14.02.2025**

Video: 3 – JavaScript, part I

Content:

Java – class-based objects with static typing.

History of Java and its beginnings.

Usage of JavaScript (included in HTML or as external file).

Basics of JS (syntax, functions, variables, commands)

JS variables var, const, let.

Loops for and while, their structure.

Operations and operators in JS.

Arrays – dynamic and objects.

Document object model.

Usage od DOM and DOM-specific methods.

onclick()

**15.02.2025**

Video: 4 – CSS, part I

Content:

CSS definition and history, main idea – separating page style from page structure.

Ways of using CSS.

CSS basics – fundamentals, comments, syntax.

Colors in CSS.

Box model in CSS, default values, features.

Px, pt, pc, in and em, ex, rem, % as measures in CSS.

Styles of specific elements.

Inlining and blocking elements.

Placing elements.

Text properties and formatting options

**16.02.2025**

Video: 5 – CSS, part II

Content:

Different ways to install styles, locate elements.

Classes and pseudo classes.

Selectors, CSS pattern.

Background formatting.

Shapes, shades in CSS.

Element placement.

Flexbox.

Libraries and preprocessors.

**19.02.2025**

Video: 6 – HTML+CSS+JS demo

Content:

Learning how to combine HTML, CSS and JavaScript in practice.

**20.02.2025**

Video: 7 – CSS, part III – responsive design

Content:

Need for use responsive design.

CSS media queries.

Layout for different devises.

Positioning, layout build.

**21.02.2025**

Video: 8 – JSON & AJAX

Content:

JavaScript Object Notation – usage.

Difference between JSON and XML.

JSON in practice.

Asynchronous JS and XML meaning.

Architecture of AJAX, model.

AJAX in practice.

**22.02.2025**

Video: 9 – Materialize.

Content:

Usage – libraries.

Grid system meaning.

Materialize for different screen sizes.

Responsive content.

Components.

**25.02.2025**

Video: 10 – Node.js & Express

Content:

Difference between Frontend and Backend.

Node.js – importance, usage.

Characteristics in node.js.

Event loop.

Theory and promises.

Package management.

Express.js.

Templates vs API.

CRUD.

**26.02.2025**

Video: 11 – MongoDB

Content:

Databases meaning.

MongoDB definition, construction.

Differences between MongoDB and SQL database.

Mongoose importance.

**27.02.2025**

Video: 12 – Docker Containers

Content:

History.

Comparison: psychical server, various apps, virtual server, docker.

Why docker – benefits.

Installation and dockfile.

Docker notes.

Containers management.

**28.02.2025**

Video: 13 – Template engine.

Content:

HTML template engine, basics, the idea.

Practice.

Model-View-Controller – meaning.

**3.03.2025**

Video: 14 – Authentication and authorization.

Content:

Authentication, why, usage.

User profile.

Validation and sanitizing.

Authentication – login and password

Authorization – right

JSON Web Token – JWT.

**9.03.2025**

Video: 15 - Web storage

Content:

Web storage meaning.

Local and session storage.

Usage of storages.

Example.

**11.03.2025**

Video: 16 – React

Content:

React history and usage.

React architecture (JSX), components.

**12.03.2025**

Video: 17 – React, part II

Content:

React fragments – components need to return always only one element.

Router (react doesn’t have routing in core).

Higher-order components (HOC).

Internationalization – i18n (various languages).

useEffect when there is a need to so sth before the page is rendered.

**14.03.2025**

Video: 18 - Fullstack

Content:

Fullstack meaning.

React and Node.js.

Example

**15.03.2025**

Video: 19 – Canvas & HTML5 games

Content:

Canvas element.

HTML5 games meaning.

Phaser benefits, usage.

Features.

Practical application.

**16.03.2025**

Video: 20 – Hacks on top of hacks.

**17.03.2025**

Video: 21 – TypeScript

Content:

TypeScript history, meaning.

Features of TypeScript.

Functions, types, parameters.

Type aliases and interfaces.

Tricks and properties.

Usage of TypeScript.

Practical application.

**Part II – exercise project**

**19.03.2025**

I choosed visual studio code as my code-editor

I made new folder where I started my project and connected repository on GitHub.

# Base HTML and Links – push 1

Basic HTML Page Structure

I created a basic HTML web page. While working on this code, I learned a few key elements of page structure and how to properly embed different resources.

Basic HTML Structure

I used <!DOCTYPE html> to specify the HTML5 version.

I added a <head> section where I placed metadata and links to external resources.

In the <body> section, I placed the <h1>Hello!</h1> heading.

Encoding Settings and Responsiveness

I used <meta charset="UTF-8"> to handle various special characters.

<meta name="viewport" content="width=device-width, initial-scale=1.0"> makes the page responsive on different devices.

Adding External Resources:

Google Fonts – I imported the Poppins font using link.

FontAwesome Icons – I added a set of icons, which will allow me to use them in the interface later.

CSS Style – I linked the style.css stylesheet to style the page.

Favicon – I added a page icon via link rel="shortcut icon" href="images/favicon.ico"

Linking to the JavaScript file:

In the <body> section, I added <script src="main.js"></script>, which allows me to add interactivity using JavaScript.

****

# Page structure extension, adding navigation – Push 2

I extended the basic HTML page by adding navigation.

I created <nav class="navbar"> to place the navigation menu. I used <div class="container"> to improve the organization of elements. I added <div class="logo"> with a logo image and a link to the main page.

I added a list <ul> containing <li> elements with links to different sections. Each <a> link leads to the appropriate page (e.g. index.html, # for sections not yet ready). I added a special login button <a class="btn btn-light"> with a user icon.

I used <i class="fas fa-user"></i> to add a user icon to the login button.

The code structure is clear, elements are well organized in containers. Using class="navbar", class="main-menu" allows for easier styling in CSS.

****

**20.03.2025**

# Basic CSS styles – PUSH 3

I added a style.css file that contains basic styling rules for the page and makes it easier to be responsive.

First, I used box-sizing: border-box, which allows you to control the dimensions of elements, eliminating the problem of unexpected margins and padding. Additionally, I set margin: 0 and padding: 0 for all elements, which helps avoid default spacing.

For the body, I set the font to Poppins, the default text size to 16px and line-height: 1.5 to increase the readability of the content. I also set the background color to white.

I styled the links so that they do not have underlining, which improves the aesthetics of the page.

To remove the default list tags, I used list-style: none for ul. This means that the lists do not have dots or other tags.

I used max-width: 100% for images, which ensures their responsiveness and adaptation to the width of the container.

I also added helper classes for containers.

.container has a maximum width of 1100px, and .container-sm has a maximum width of 800px.

Both classes are auto-centered with margin: 0 auto and inner padding of 0 15px, which helps maintain proper margins on different screens.

# Extending CSS styles – variables, navigation, buttons – PUSH 4

I added new elements to the style.css file, extending its functionality.

I introduced CSS variables using :root, which allows for easier management of the page's color scheme. I defined:

--primary-color: #4891ff; – main page color

--light-color: #f4f4f6; – light secondary color

--dark-color: #111; – dark color of text and elements

I added styling for the navigation (.navbar). I gave it a white background and 20px padding. Thanks to display: flex in .navbar .container the menu was set in one line and the elements are evenly spaced.

I set the .navbar .main-menu ul list as flexbox, which allows for displaying links horizontally. Navigation links have received 10px 20px padding, and the hover effect changes their color to var(--primary-color).

An icon has also been added to the .navbar ul li a i links, with a 10px margin on the right side.

I have extended the utility classes, adding .container and .container-sm to better manage the width of the content.

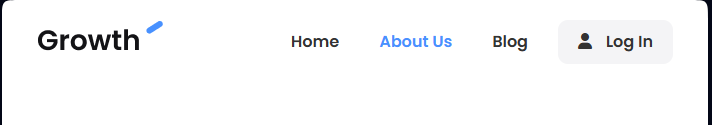
I have added styles for buttons (.btn). The buttons have:

Default background var(--light-color), rounded corners (border-radius: 10px)

Hover effect (opacity: 0.8)

Color variants .btn-primary, .btn-dark and .btn-block for different applications.

These changes improve the aesthetics and structure of the page, and the variables make it easier to further edit the color scheme.



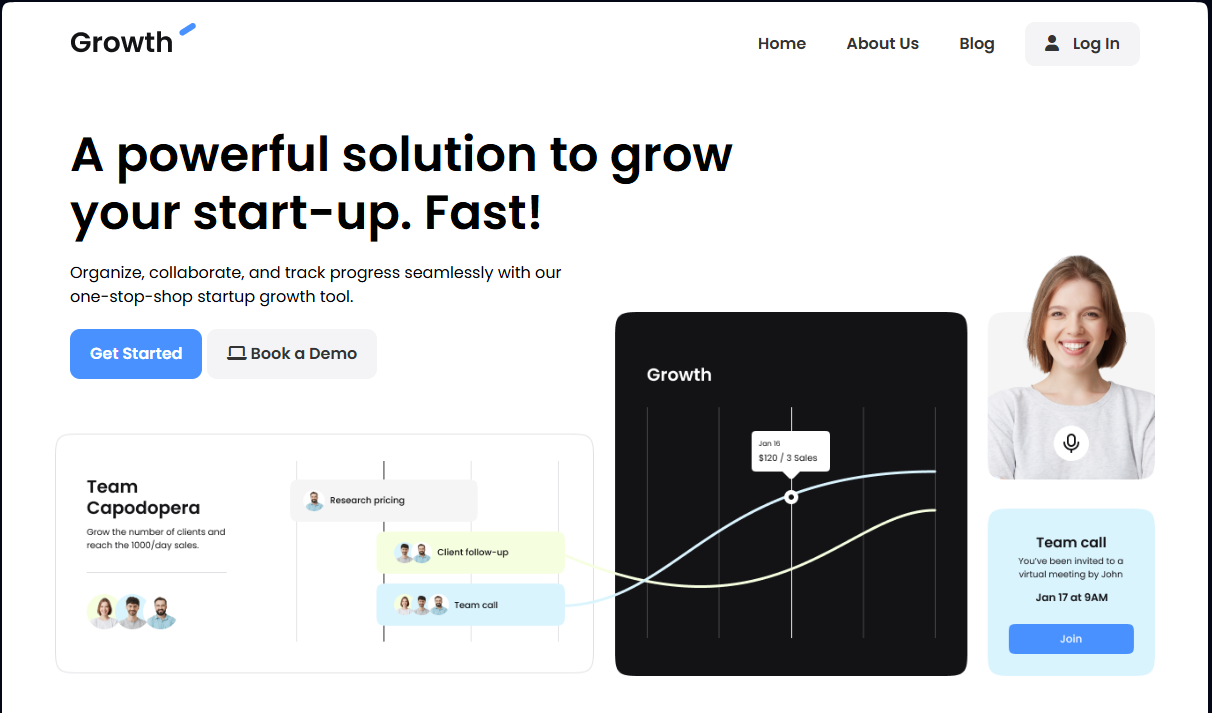
# Hero and utility classes – PUSH 5

I added a new hero section, which introduces the main header and call-to-action buttons.

In HTML, I added a section that contains a header, description, and two buttons: one to start using the app, the other to schedule a demo.

In CSS, I added styles for the hero section, including background, sizes, and content layout.

I also added text classes, which allow for easy management of the size and styling of headers and text on the page.



# Video section – PUSH 6

Video section in HTML

I've added a new Video section that presents a video preview and a button encouraging the user to start using the application. The section contains:

A h2 header encouraging them to watch the presentation,

A preview image of the video,

A Get Started button.

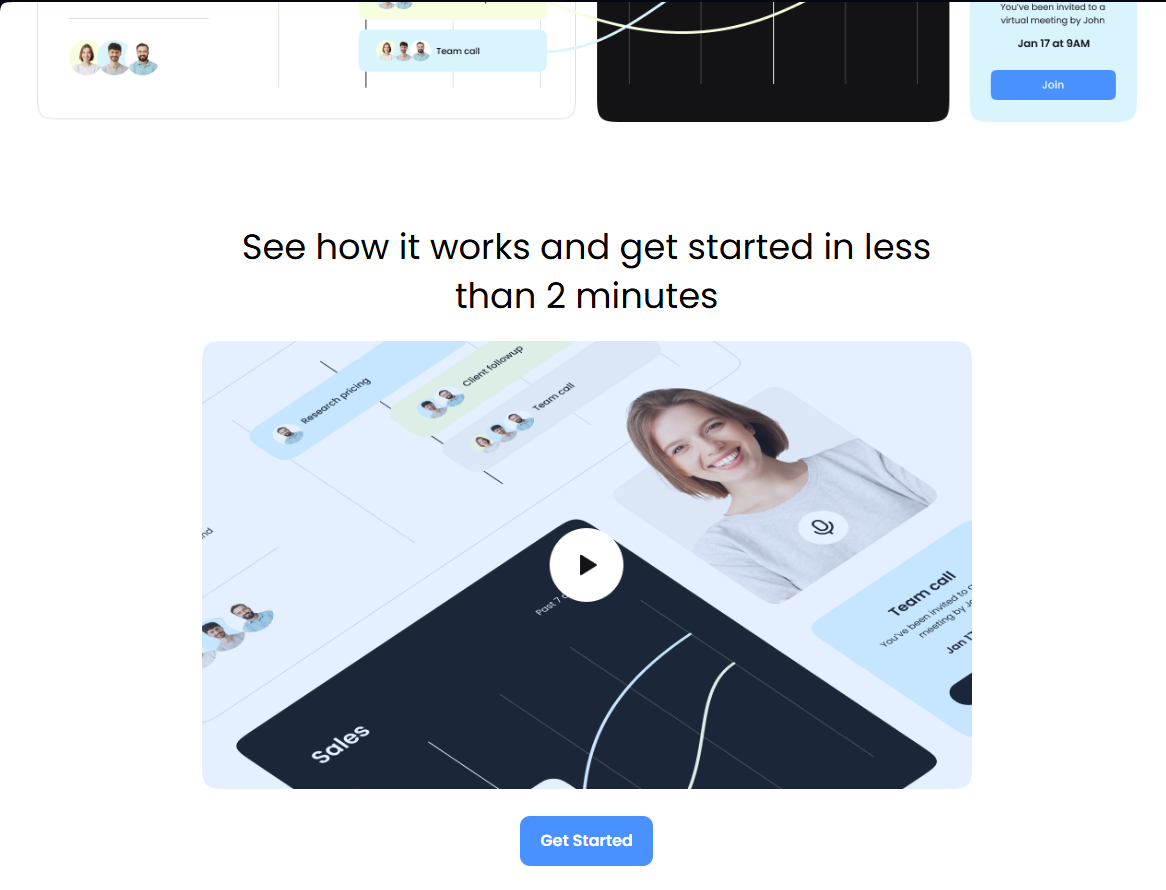
Styles for the Video section in CSS

To properly display the Video section, I've added new styles:

.video - defines the appearance of the section, including the background and margins,

.video .video-content - sets the layout of elements in a column and centers the content,

.video .video-preview - adds a space between the image and the button.



# New Background section in CSS – PUSH 7

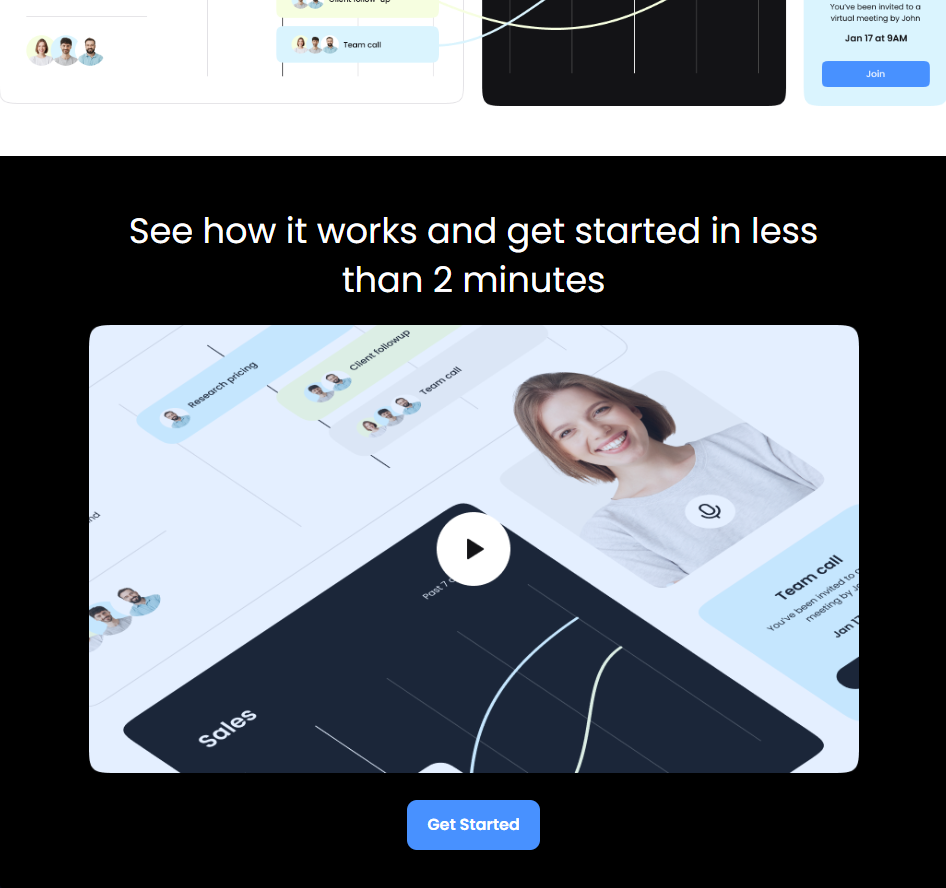
I've added new classes responsible for backgrounds for different sections of the page. Here are their functions:

.bg-primary - sets the background to the primary color (--primary-color) and changes the text color to white,

.bg-light - sets the light background (--light-color) and the text color to dark (#333),

.bg-dark - sets the dark background (--dark-color) and the text color to white,

.bg-black - sets the black background and the text color to white.



# Testimonials – PUSH 8

Changes made to HTML

Added Testimonials section (<section class="testimonials bg-dark">)

The new section includes a header:

"Don't just take our word for it, see the success stories from businesses just like yours."

Added three customer testimonial cards (<div class="card">), each with a short quote and the author's signature.

Added .bg-dark class to Testimonials section

This makes the background of this section dark (var(--dark-color)) and the text white, which improves contrast and readability.

Changes made in CSS

1. Added Testimonials section (.testimonials)

New .testimonials class added to styling of customer testimonials section.

Styling:

Background set to dark (.bg-dark).

Centered text (text-align: center).

Added top and bottom padding.

2. Added styling for cards (.card)

New .card class to format user testimonials.

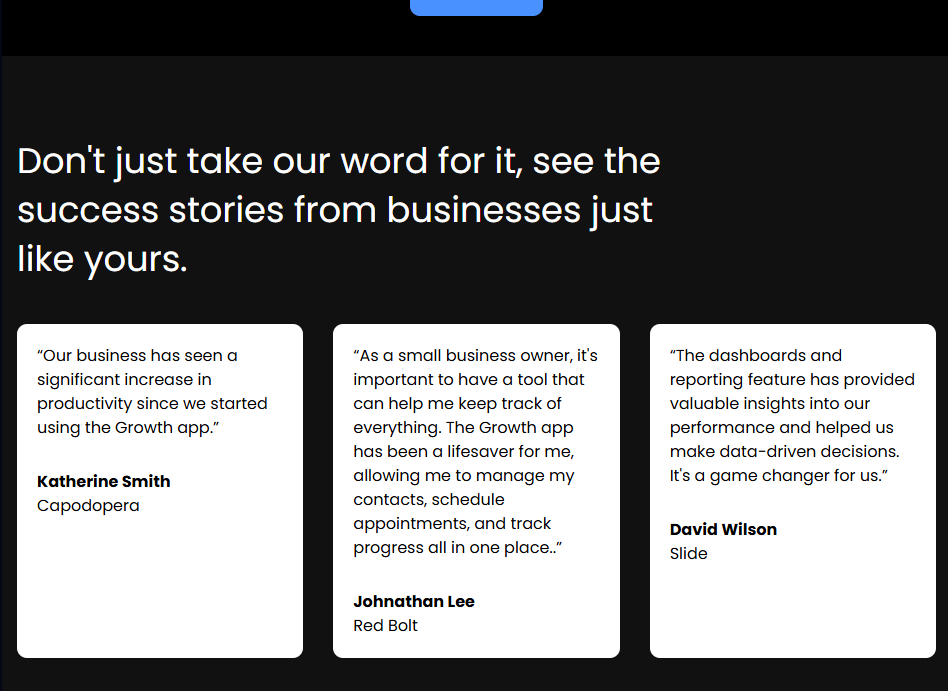
Styling:

Background: light (var(--light-color)) for better contrast.

Rounded corners: more aesthetic finish.

Drop shadow: adds a card lift effect.

Spacing: improves readability and layout.



# Pricing section is HTML file and CSS file – PUSH 9

Added "Pricing" section in HTML file:

I added a Pricing section to the page to display information about the subscription plans offered by the app.

The section contained a header, subheading, and two cards with plans: one free ("Simple") and one paid ("Premium").

In both cards, I included brief descriptions of the plans, prices, and a list of features available in the plan. In the Premium plan, I added a monthly price ($49) and additional features available in that plan.

CSS changes:

Styling of the "Pricing" section:

I added styles for pricing-grid to display the cards in a grid (2 columns), with appropriate space between them.

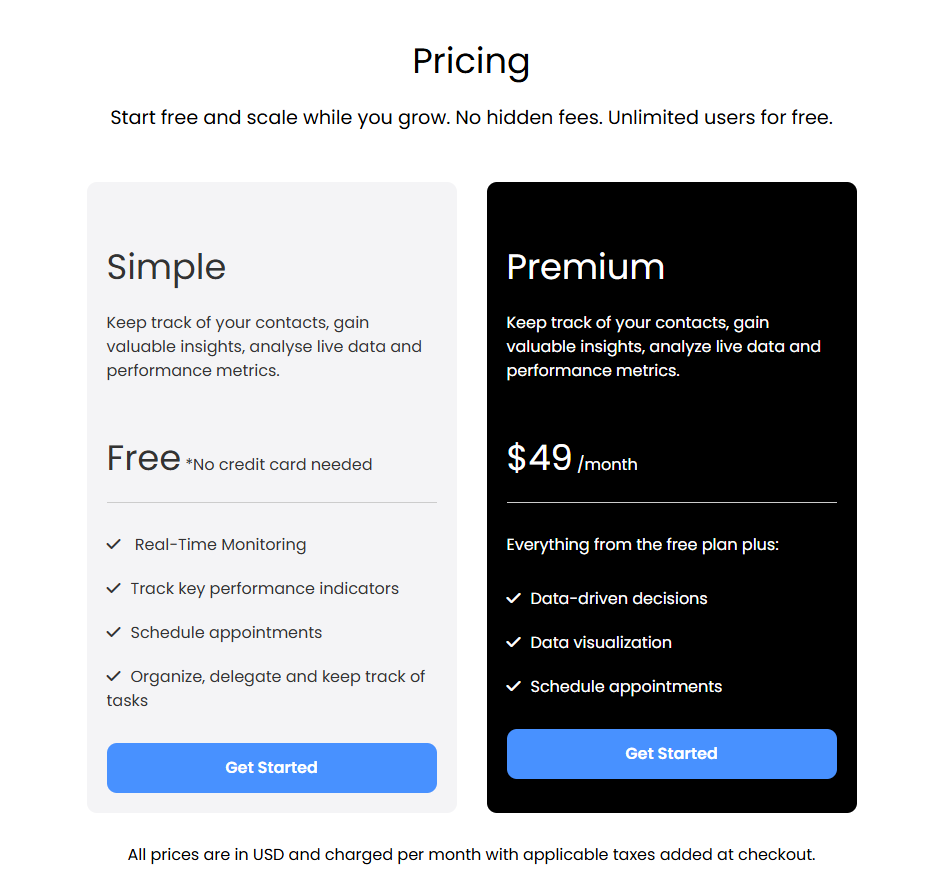
I adjusted the styling of the cards, adding a background, margins, and rounded corners to make the cards more aesthetically pleasing and cleaner.

The styling of the buttons in the cards was set to full width (btn-block) and changed the background color of the buttons to var(--primary-color).

Styling of titles, descriptions and prices:

I added styles for section headings (pricing-heading, pricing-subheading) to make them stand out properly.

I adjusted the appearance of prices, using the appropriate spacing and adding an underline for the price in the Premium plan.



# FAQ SECTION – PUSH 10

Added "FAQ" section in HTML file:

I added a "FAQ" section to the page that contains the most frequently asked questions about the application.

The section consists of a header, and below it are questions and answers in the form of expandable panels (accordion).

Each question is clicked to display a detailed answer. This makes navigation easier and the page more interactive.

CSS changes: Styling of the "FAQ" section:

I applied styling to the expandable panels (accordion).

I added appropriate space between questions and rounded corners to each panel to make it look aesthetically pleasing.

The question headers have been highlighted in bold and the font size has been adjusted to be visible.

The styling of the expandable buttons has been adjusted to match the overall aesthetic of the page, with appropriate background colors and margins.

Added "FAQ Accordion" functionality in JavaScript:

I've added an expandable panel (accordion) feature to the FAQ section. The JavaScript code allows you to interactively expand and collapse answers to questions. When you click on a question header, the icon changes from plus to minus, and the answer text becomes visible

The functionality also closes other expanded answers when the user opens a new one, so only one answer is visible at a time.

The code works by listening for clicks within the FAQ container, checking if a group header was clicked, and updating the icon classes and answer visibility accordingly.

# Footer – PUSH 11

Added "Footer" section in HTML file:

Footer consists of the following elements:

* Company logo, which is a link to the home page.
* Subscription form, allowing users to enter their email address to receive business advice.
* Social media icons with links to LinkedIn and Twitter.
* Sections with company information (About Us, Our Process, Join Us), resources (News, Research, Recent Projects), and contact information (email address).

CSS changes: Styling of "Footer" section:

To properly style the footer, I made the following changes:

Dark background (#000) for the footer to match the overall design of the page.

Each footer section (company, resources, contact) is displayed in a grid layout, which ensures even content distribution and easy readability.

The logo is placed in the top left corner with appropriate padding, and the subscription form and social media icons are placed below.

The subscription form has an email address input field and a button to submit the form, which are styled to resemble buttons on the page.

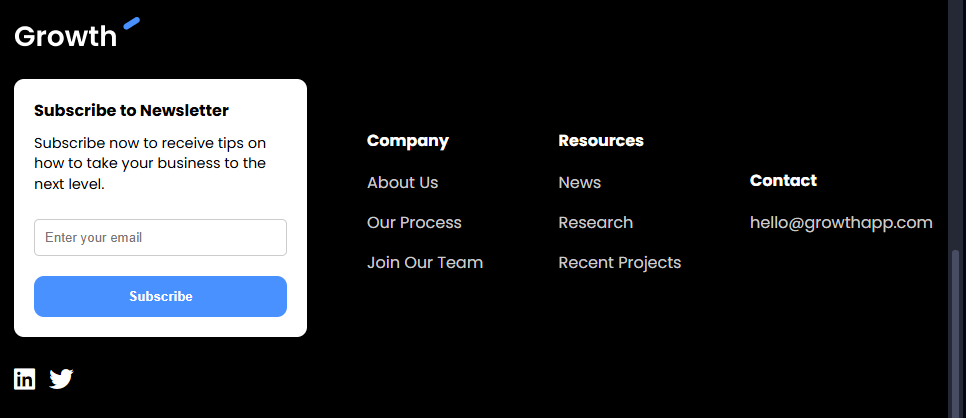
I added hover effects to the social media icons, changing their opacity on hover, which makes the page more interactive.

Footer Responsiveness:

The footer has been designed to be responsive, meaning it adapts to different screen sizes. The grid layout ensures that the footer content will be well-spaced on both large and small screens, and the form sections and other footer elements will stack vertically on smaller devices for easy accessibility.

JavaScript Functionality:

No additional JavaScript functionality has been added to the footer section, as it contains mostly static content. However, the subscription form can be enhanced with a validation script or AJAX functionality to improve the user experience when using the form.



# Mobile mEnu HTML

1. Adding a hamburger button: To make navigation easier on mobile devices, I added a hamburger button to the navigation menu.

HTML code:

<button class="hamburger-button">

<div class="hamburger-line"></div>

<div class="hamburger-line"></div>

<div class="hamburger-line"></div>

</button>

2. Mobile menu expansion feature: The navigation menu that is displayed when clicking the hamburger button is placed in the <div class="mobile-menu"> element. This is a version of the menu that is adapted to display on mobile devices. It contains the same menu items as the main version, but in a more compact and easy-to-use format.

HTML code:

<div class="mobile-menu">

<ul>

<li><a href="index.html">Home</a></li>

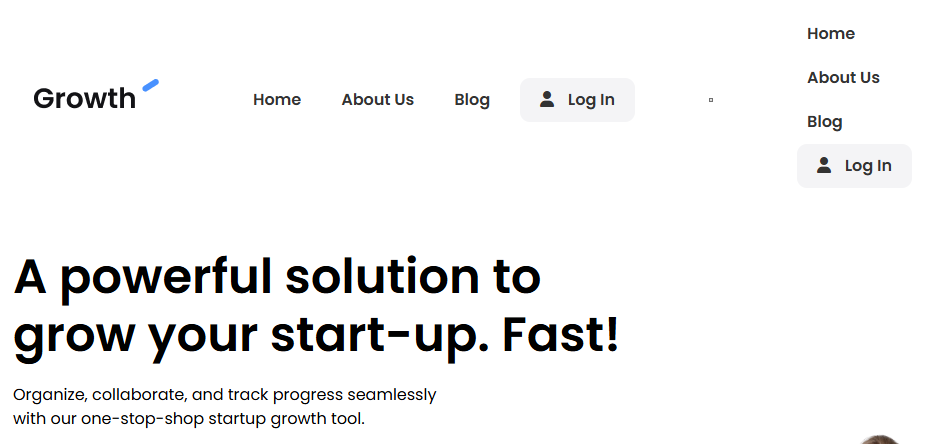
<li><a href="#">About Us</a></li>

<li><a href="#">Blog</a></li>

<li><a class="btn" href="#"><i class="fas fa-user"></i> Log In </a></li>

</ul>

</div>



# Mobile mEnu CSS

CSS changes that fix the double menu issue on desktop and improve mobile navigation:

Hamburger button:

The hamburger button is initially hidden on larger screens (above 670px) with display: none.

It is only displayed on smaller screens (below 670px) and is used to display the mobile menu.

The hamburger button consists of three horizontal lines (.hamburger-line) that form the distinctive menu button. The lines are 30px wide, 3px high, and have a background color of #333. They are evenly spaced with a margin of 6px between them.

Mobile menu:

The mobile menu is a persistent menu that initially sits off-screen. It is hidden with right: -300px to make it invisible.

When the menu becomes active (.mobile-menu.active), it slides into the screen from the right side with a smooth transition effect, using transition: right 0.3s ease-in-out.

The menu is 250px wide and takes up the full height of the screen.

The menu contains lists of items that have 10px spacing between them. Links have a transition effect for smooth interaction.

Media Queries for Responsiveness:

For screens smaller than 960px:

Reduces the font size of .text-xxl to 2.5rem to accommodate smaller screens.

For screens smaller than 670px:

The main navigation menu (.navbar .main-menu) is hidden with display: none, allowing only the hamburger button to be displayed.

The hamburger button (.navbar .hamburger-button) becomes visible, allowing the user to open the mobile menu.

The background of the hero section is adjusted for smaller screens.

The content in sections such as .hero .hero-content and .hero .hero-text is aligned and adjusted for smaller screens.

Re-layout of other sections like .testimonials, .pricing and .footer to use a single column layout (grid-template-columns: 1fr) for better readability on small screens.

Fix for double menu issue on desktop:

By using @media query (max-width: 670px), the mobile menu is only displayed on smaller screens. This hides the traditional navigation menu (.navbar .main-menu) on larger screens, so that the double menu does not appear on desktop.

In summary, the CSS changes and implementation of the hamburger button with mobile menu solve the double menu issue on desktop and provide a smooth mobile navigation experience. The layout is fully responsive, ensuring proper display on different screen sizes.

# Mobile mEnu JAVA

JavaScript changes for the mobile menu:

Listening for page load:

The script starts running when all the content of the page is loaded, using document.addEventListener('DOMContentLoaded', ...). This ensures that all HTML and other resources on the page are already loaded before the script starts executing.

Assigning the hamburger button and mobile menu to variables:

const hamburgerButton = document.querySelector('.hamburger-button'); – This code selects the hamburger button (which the user clicks to open the mobile menu) and assigns it to the hamburgerButton variable.

const mobileMenu = document.querySelector('.mobile-menu'); – Selects the mobile menu itself and assigns it to the mobileMenu variable.

Adding a listener for the hamburger button click:

hamburgerButton.addEventListener('click', ...) – This code sets the listener for the hamburger button click. When the button is clicked, a function is executed that toggles the active class on the mobileMenu element.

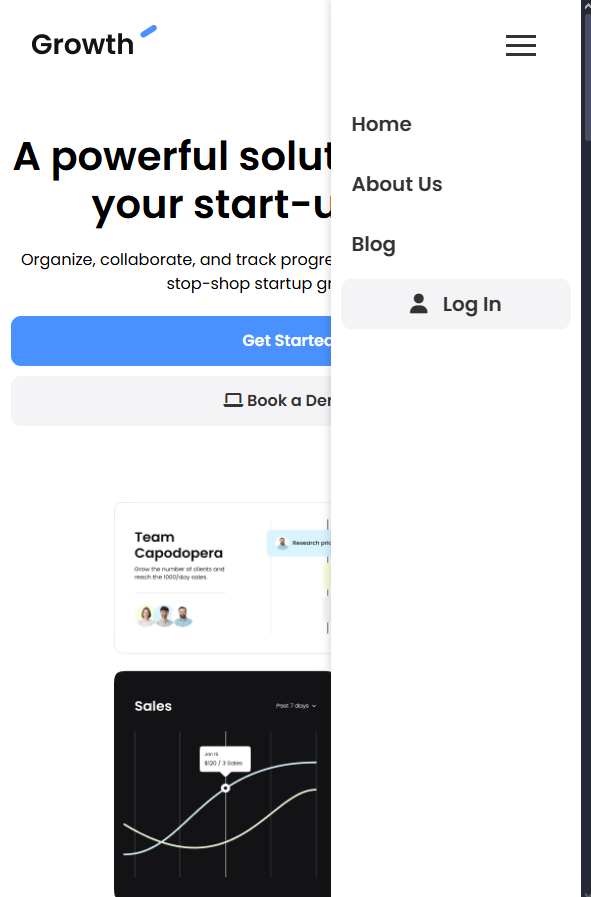
Toggling the active class on the mobile menu:

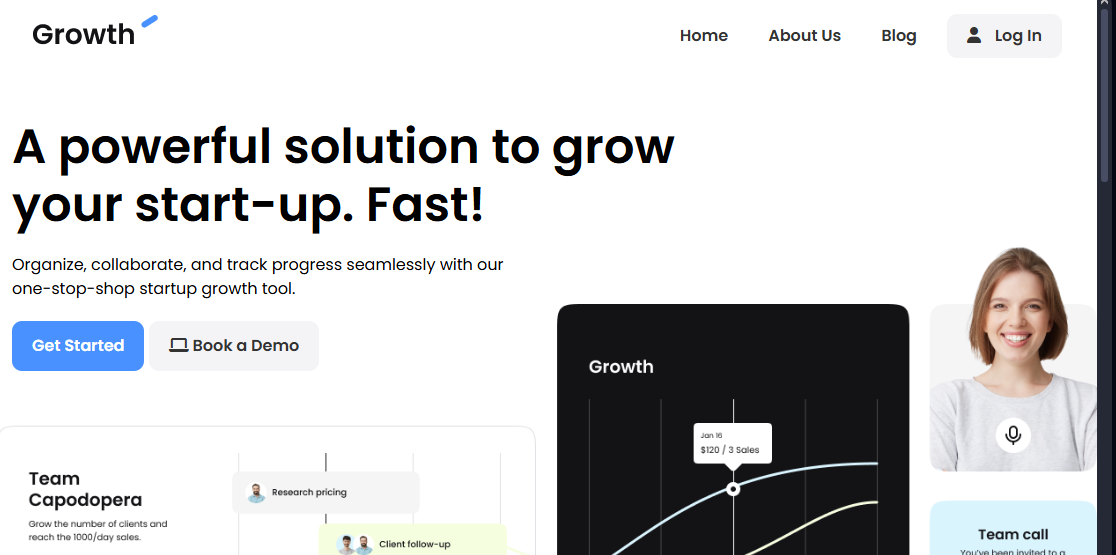
mobileMenu.classList.toggle('active') – toggle is a method that adds the active class to the menu if it is not already there, or removes it if it is already there. This way, when clicking the hamburger button, the mobile menu will appear and disappear (toggle visibility).

The result of the code:

When clicking the hamburger button, the mobile menu (specified by the .mobile-menu class) will toggle its visibility – opening and closing – on smaller screens. This is a key element to provide interactivity in the mobile version of the page.

The script allows to open and close the mobile menu using the hamburger button. Using the active class provides a smooth transition between hidden and visible states, which improves the usability of the page on mobile devices.





# Full code

HTML structure description of the "Growth App" page:

Header (<head>):

The page uses several external resources, such as fonts from Google Fonts (Poppins) and icons from Font Awesome.

Introduces links to CSS sheets, both to external ones (Google Fonts, Font Awesome) and to the local style.css file, which contains the overall style of the page.

A favicon (page icon) and responsiveness settings (meta viewport) have also been added.

On the main page (<body>):

On the navigation:

The menu contains the logo, standard links to pages, and a hamburger button, which is visible on mobile devices.

The mobile menu is displayed after clicking the hamburger button, which allows for easier navigation on smaller screens.

"Hero" section (introduction to the page):

Large, prominent headers and buttons that encourage you to start using the app or book a demo. This is a typical element that attracts the attention of visitors.

Video section:

A short video introduction that helps users understand how the app works. It also contains a button that redirects to the registration page.

User reviews section (Testimonials):

This section contains reviews from satisfied app users, which adds credibility to the site.

Pricing section:

It presents the available pricing plans of the app: a free plan and a premium plan. Each plan is described with information about the features and price.

FAQ:

A list of frequently asked questions that help users clear their doubts and learn more about the app's functionality.

Footer:

It contains the logo, social media links, a newsletter subscription form, as well as links to information sections such as "About Us", "Our Processes", "Contact Us", etc.

Summary:

The site is responsive and adapted to various devices (including mobile devices).

The structure of the page is clear and logical, with an emphasis on clarity, encouraging the user to further interact with the app (e.g., signing up for a newsletter, registering for the app).

Features such as hamburger menu, forms, and buttons are designed in an interactive way, which improves user experience.

CSS code is responsible for styling the website, making it look great and responsive. Summary of the key elements:

Resetting margins and paddings – The global setting \* { box-sizing: border-box; margin: 0; padding: 0; } ensures that elements are uniformly sized and positioned.

CSS variables – :root defines the primary colors that can be used throughout the code, e.g. --primary-color for the primary color.

Responsiveness – Media queries adjust the appearance of the page on different devices. They change text sizes, layout of elements, and switch menus to mobile versions.

Layout – Flexbox and grid are used to arrange elements in headers, grids, and other sections, facilitating responsiveness.

Interactions and animations – Transitions on hover links and animation effects when opening mobile menus, thanks to transition.

Button and Link Styling – Define styles for buttons (.btn), including colors, backgrounds, and hover effects.

Mobile Menu – The mobile version of the menu is hidden by default, but appears when the user clicks the hamburger button.

In short, it's a complete set of styles that provides a beautiful and functional interface adapting it to different resolutions and devices.

JavaScript code contains two main functionalities:

FAQ Accordion: After the page loads, it adds the function of expanding and collapsing the FAQ section. When the user clicks on the group header, it expands and the icon changes from "+" to "-" (or vice versa). Only one FAQ section can be open at a time - the others will be closed.

Mobile Menu: After clicking the hamburger button (menu icon), the mobile menu is shown or hidden by toggling the active class on the menu item.

Both functions are activated after the page is fully loaded (DOMContentLoaded).

From this project I learned several important things, both in terms of design and implementation of functions in JavaScript.

* Creating responsive websites: I learned how to adapt the layout to different screen sizes using media queries, which allows you to create a page that looks good on both computers and mobile devices.
* Working with Flexbox and Grid: Using Flexbox and CSS Grid, I learned how to easily organize the layout of elements on the page, creating flexible and responsive structures.
* Styling interactive components: I mastered techniques for creating interactive elements such as buttons, accordions (FAQ) and drop-down menus that respond to user interactions, thus improving the experience on the page.
* Working with JavaScript: I implemented functionalities in JavaScript, such as drop-down menus (hamburger menu) and FAQ accordion. I learned how to respond to user events, manipulate CSS classes and toggle the states of elements on the page.
* Managing HTML document structure and organizing code: I understood how important it is to organize HTML, CSS, and JavaScript properly to make the code readable and easy to maintain.
* Using CSS variables: I used CSS variables to simplify color and style management, which allowed for easier changes to the look of the page in the future.
* Mobile optimization: Thanks to the project, I learned how to customize elements of the page to be mobile-friendly, such as by changing text sizes, layout of elements, and adjusting the background.

This project gave me a better understanding of how important it is to combine HTML, CSS, and JavaScript.

Project is deployed with GitHub Pages.

<https://github.com/koszmary/Front-end_2025>

<https://koszmary.github.io/Front-end_2025/>

**Part III – course project**

## 21.03.2025

## Concept:

I couldn't find an idea for the project for a long time, but since I love traveling, I decided I wanted to create a chain of hotels. The idea came from a business class we had in high school. I called it "Mway" because it sounds modern.

I wanted the website to be aesthetic, functional and user-friendly.

Here's how I came up with it:

Homepage:

At the very top there should be a large, eye-catching banner that will immediately show the character of the hotels - elegant, but with a hint of adventure. Below the banner I want to place a few reviews from guests because I think it adds credibility to the website and shows that the hotels are really unique, and I liked this idea from the template.

"Our hotels" tab:

Here users will be able to see in which countries my hotels are located. After clicking on the selected country, a list of hotels with short descriptions, photos and the most important information will expand. I want it to be interactive and easy to use, so I plan to use the JavaScript code from the template.

"Our Story" tab:

This is where I tell how the Mway network was created. I want there to be photos showing the development of the project. The photos should be responsive, so they look good on every device.

"Contact Us" tab:

Since I often use contact forms on other websites, I thought it was a must-have. I want guests to be able to easily send a message, ask about an offer or simply share their impressions, and then I'll also think of something to add responsiveness.

Footer:

At the bottom of the page, there should be a footer that will sum up the whole thing. I like the one from the template, so I'll probably use it. There will be links to social media, contact details and maybe some additional information.

The website should be modern, but also friendly and encouraging to discover hotels. For now, I want it to be maintained in a navy blue, white and blue aesthetic, but we'll see what happens.

INDEX.HTML – home page:

In the project I use a template to make a header and footer, I only change the colours and names. I left the fonts. To make the graphics for the website I used the Canva website for the logo and the rest using Gimp. To make the card icon I used some random generator.

Step 1: HTML Document Structure

I started by creating the basic structure of my HTML document, using <!DOCTYPE html> to declare the document type as HTML5.

In the <head> section, I added metadata such as charset="UTF-8" to display characters correctly and a viewport to make the page responsive on different devices.

I set the page title to "Mway - Home" which will be visible in the browser tab.

Step 2: Adding Fonts and Icons

I used Google Fonts to add the Poppins font because they were cool.

I also added the Font Awesome icon library via a CDN so I could use icons in different sections of the page.

Step 3: Linking CSS and Favicon Files

I linked an external CSS file (style.css) which contains the styles for the entire page which I will deal with later.

I added a favicon (page icon) which appears in the browser tab to make the page look more professional.

Step 4: Creating navigation (Navbar) - based on the template

I created a <nav> section with the navbar class, which contains the site logo and the navigation menu.

The navigation menu consists of links to the pages: Home, About Us, Contact Us and Hotels.

I added a hamburger button with three lines, which will be visible on mobile devices. After clicking, the mobile menu will expand, but that will be later, when I create a java file (I will take it from the template).

Step 5: Hero section (Main banner)

In the <section class="hero"> section, I placed a large banner that attracts the user's attention.

I added a heading (h1) with an encouraging text: "Experience Luxury, Embrace Comfort - Your Perfect Getaway Awaits!".

Below the heading, I placed a short description (p) and three buttons (btn), which lead to different sections or hotel offers.

Step 6: Guest Testimonials Section - from the template

I created a <section class="testimonials"> section, where I placed guest testimonials.

Each testimonial is presented as a card, containing the text of the testimonial, the guest's name and their status (e.g. Frequent Traveler).

The section has a bg-primary background color to stand out from the rest of the page.

Step 7: Footer

In the <footer class="footer"> section, I placed the site logo, newsletter subscription form, and social media links.

I added sections with information about hotels, services, and contact details.

At the very bottom of the footer, I placed a copyright notice and links to the privacy policy and terms of use.

Step 8: Responsiveness and JavaScript

I connected a JavaScript file (main.js), which will be responsible for handling interactions, such as expanding the mobile menu.

I checked that all links work correctly and the forms are functional.

Next I done HTML pages for other tabs.

For About US tab:

Step 1: Using the Previous HTML

I used the HTML I had already created, which already included a navbar and footer. This saved me time and allowed me to focus on adding a new About Us section and gallery.

Step 2: Adding Text with ChatGPT

I used ChatGPT to generate the text for the About Us section. The text is consistent and professional, which adds credibility to the page.

The About Us section contains three subsections: Our Story, Our Mission, and Our Team.

Step 3: Finding Images for the Gallery

I used various websites to find commercially available images that fit the brand image.

Each image has a short description.

Step 4: Creating the About Us Section

In the <section class="about"> section, I included information about the history, mission, and team of Mway Hotels.

I added the About Us heading and three subsections: Our Story, Our Mission, and Our Team.

Step 5: Creating a Gallery Section

In the <section class="gallery"> section, I placed a gallery of photos of hotels.

The photos will be responsive.

For Contact us:

Step 1: Using the Previous HTML

I used the HTML I created earlier, which already included a top bar (navbar) and a footer (footer).

The change was that I made a separate css file for this subpage.

Step 2: Adding the Contact section

In the <section class="contact"> section, I placed contact information and a form to send a message.

The Contact section contains three subsections with contact information for hotels in Norway, Sweden, and Finland. Each subsection contains a phone number, email address, and the manager's name and surname. I'll make it responsive later.

I added a contact form that will allow users to send a message to the selected hotel. The form contains fields to select a country, hotel, and enter a message.

For Hotels subpage:

Step 1: Using the Previous HTML

I used the HTML that I created earlier, which already included a top bar (navbar) and a footer (footer).

Step 2: Adding the Our Hotels section

In the <section class="hotels"> section, I put information about hotels in Norway, Sweden, and Finland.

The Our Hotels section contains three subsections: Norway, Sweden, and Finland. Each of these contains a list of hotels with contact information such as address, phone number, and email address.

Each subsection will expand when you click on the header to have JavaScript. I will be using the template as a reference.

CSS file:

When creating the CSS styles, I took inspiration from the template to ensure consistency and a professional look for the site. Some elements, such as the color scheme, section layout, and buttons, are almost identical to the sample designs, which allowed me to save time and focus on customizing them to my needs.

The main color scheme of the site is based on three colors: dark blue (--dark-color), light gray (--light-color), and accent blue (--primary-color). These colors are consistent throughout the design, giving the site a uniform and modern look.

The Navbar section was designed with a dark background and white text, which contrasts with the rest of the page and draws attention. Additionally, the navigation is responsive - on smaller screens, a hamburger button appears that expands the mobile menu.

The Hero section was designed to be the visual focal point of the page. This section is responsive.

The Testimonials section uses a grid layout to display guest reviews in three columns. Each review is presented in a card format, which adds elegance and order.

The Footer has also been designed with a dark background to be consistent with the rest of the page. It contains a subscription form, social media links, and contact information. The footer layout is responsive – on smaller screens, the columns are arranged one below the other.

The Hotels section has been designed with expandable sections for each country (Norway, Sweden, Finland). Each section contains a list of hotels with contact information. The expandable sections are interactive – clicking on the header expands the list of hotels.

The About Us section has been designed with a light background and dark text, which ensures readability. It contains subsections describing the hotels. The texts have been generated with ChatGPT to be consistent and professional.

The Gallery section uses a grid layout to display hotel photos. Each photo has a caption that briefly describes the hotel. The photos are responsive and adapt to different screen sizes.

All sections are responsive, meaning the site looks good on a variety of devices, from desktop to mobile. Media queries were used to adjust the layout and font sizes on smaller screens.

Additionally, the site features animations, such as smooth section expansion in the Hotels tab and hover effects on buttons and images, which add interactivity and improve the user experience.

Overall, the site was designed with consistency, responsiveness, and ease of navigation in mind. Using ready-made solutions from templates allowed me to focus on customizing the styles to my needs while maintaining a professional look.

CSS file for contact tab:

When creating the CSS styles, I took a cue from an earlier CSS file that already contained the basic styles for the page. This allowed me to maintain visual consistency and focus on customizing and expanding the existing sections.

The Contact section was added as a new feature. It contains contact information for hotels in Norway, Sweden, and Finland, as well as a form for sending messages. The form is responsive, and on smaller screens, the contact cards are arranged one under the other.

JS file:

When creating JavaScript functionality, I took inspiration from existing solutions to ensure interactivity and smooth operation of the site. Some elements, such as drop-down FAQ sections or mobile menus, are based on ready-made examples, which allowed me to save time and focus on adapting them to my needs.

1. Drop-down FAQ sections (Accordion) - from a template

I added functionality for drop-down FAQ sections, which allows users to click on the header to show or hide the content. I used the click event on the .faq-content container for this.

When clicking on the header (faq-group-header), the icon changes from fa-plus to fa-minus, and the content of the section (faq-group-body) is shown or hidden by adding/removing the open class.

Additionally, if one section is open and the user clicks on another, the previous section automatically closes to maintain order.

2. Mobile menu - from a template

I added functionality for a mobile menu, which appears after clicking the hamburger button. I used the click event on the .hamburger-button button for this purpose.

After clicking the button, the mobile menu (mobile-menu) moves to the screen, and after clicking again – hides. This effect is achieved by adding/removing the active class.

3. Contact form

I added dynamic filling of the list of hotels in the contact form based on the selected country. I used the change event on the country selection field (#country) for this purpose.

After selecting the country, the list of hotels (#hotel) is automatically filled with the appropriate options. Hotel data is stored in the hotels object, where each country has an assigned list of hotels.

The contact form (#contactForm) also contains validation – if the user does not fill in all the fields, an alert is displayed asking to fill in the missing information.

After the form is correctly filled in, an alert is displayed with confirmation of sending the message, and the form is reset.

Link for website: <https://koszmary.github.io/Front-end_2025/PROJECT/>