Comprehensive Exercise Report

Team "The Frenchies" of Section 392

names :
Tristan Guy Michel Perrot 250ADB057
Gabriel Antoine Maurice Decuyper 250ADB059
Amaury Guy Eugène Duplat 250ADB060
Mathis Braux 250AEB032
Antoine Breteau 250AEB033
Antoine Dieudonné 250ADB058
Thomas Olivier Sylvain Finkelstein 250ADB061

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/-analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
  - **One sentence description:** The project is to develop a professional, single-page website for Matéo Ginisty to showcase his high-end car collection curation and management services, enabling potential clients to understand his offerings and contact him.
  - **Known requirements from client description:**
    * The website must present Matéo Ginisty's specialized services in private automobile collection curation and management.
    * It should target members of automobile clubs (like Automobile Club de Genève) and private collectors.
    * The site must convey professionalism, discretion, and a personalized approach.
    * Services to be detailed include:
      · Integral Collection Management (inventory, maintenance planning, general condition monitoring).
      · Maintenance and Preservation (professional cleaning, secure storage, pre-use preparation, regular reports).
      · Logistics and Transport (secure transport, administrative assistance for international travel).
      · Acquisition and Cession Advice (personalized search, professional evaluation, negotiation support).
    * The site should highlight benefits for clients: serenity, time-saving, heritage preservation, access to a professional network.
    * Contact information (name, phone, email) must be clearly available.
    * The website should have a sophisticated and premium feel.
- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
  - **Questions and instructor's answers:**
    * **Q1:** Are there any specific branding guidelines (logo, specific color palette beyond a general "premium" feel, typography preferences beyond "Oswald")?
      · **A1 (Instructor):** The client is open to suggestions but likes the current dark theme with the Oswald font. A simple text-based logo for "MATÉO GINISTY" is acceptable for now.
    * **Q2:** Does the client have preferred imagery (e.g., specific types of cars, style of photography) or will stock/provided images be used?
      · **A2 (Instructor):** The provided 'index.jpg', 'section1.jpg', 'section2.jpg' are representative of the desired aesthetic.
    * **Q3:** Is there a preference for how contact form submissions are handled (e.g., email to a specific address, integration with a CRM)?
      · **A3 (Instructor):** For now, submissions should be directed to 'mateogin-

isty@hotmail.com‘ (as per client doc). Formspree is an acceptable solution.
    * **Q4:** Are there any specific competitors whose websites Matéo Ginisty admires or dislikes?
        · **A4 (Instructor):** No specific competitors mentioned, focus on a unique, high-end presentation.
    * **Q5:** Beyond the services listed, is there any other content to include (e.g., a more detailed "About Me" section, client testimonials)?
        · **A5 (Instructor):** The current "Mon approche" section serves as an introduction. Testimonials can be considered for a future iteration.
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  - Specifics of high-end car collection management (general understanding is sufficient for the website).
  - Advanced CSS ‘backdrop-filter‘ for "glassmorphism" effect (Ref: MDN Web Docs).
  - JavaScript scroll-triggered animations and background transitions (Ref: MDN Web Docs on ‘scroll‘ events, ‘requestAnimationFrame‘, CSS Transitions/Animations).
  - Formspree integration (Ref: Formspree.io documentation).
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  - High-net-worth individuals who own or wish to build valuable car collections.
  - Members of exclusive automobile clubs.
  - Busy professionals or enthusiasts who require expert assistance in managing their automotive assets.
  - They are likely technologically literate, expect a high-quality user experience, and value privacy and discretion.
- Describe how each user would interact with the software
  - Users will navigate the single-page website by scrolling. They will read about Matéo Ginisty's approach and services. If interested, they will use the contact form to submit an inquiry, providing their contact details and specifying their needs. They expect a seamless, visually appealing experience on both desktop and mobile devices.
- What features must the software have? What should the users be able to do?
  - Display a compelling hero section.
  - Clearly present Matéo Ginisty's unique approach.
  - Detail the range of services offered with concise descriptions.
  - Provide an easy-to-use contact form to request information or a consultation.
  - The website must be responsive, adapting to desktop, tablet, and mobile screen sizes.
  - Visually engaging design with smooth transitions and animations.
  - Display contact information (phone number in footer).
- Other notes:
  - The overall tone of the website should be one of exclusivity, trustworthiness, and expert knowledge. The design should be clean, modern, and sophisticated. Performance (load time, smoothness of animations) is important for user experience.

Software Requirements

Use your notes from above to complete this section of the formal documentation by writing a detailed description of the project, including a paragraph overview of the project followed by a list of requirements (see lecture for format of requirements). You may also choose to include user stories.

Project Overview: The project is to develop a single-page, responsive marketing website for Matéo Ginisty, an expert in private car collection curation and management. The website will serve as a digital brochure, informing potential high-net-worth clients and automobile enthusiasts about his specialized services, unique approach, and the benefits of engaging his expertise. The primary goal is to generate leads through a contact form, while projecting an image of professionalism, luxury, and discretion. The website will feature a modern, visually engaging design with smooth navigation and animations.

Functional Requirements (FR):

- **FR1:** The system SHALL display a hero section with a primary headline ("Votre collection, mon expertise.") and a subtitle ("Profitez, je m'occupe du reste.").
- **FR2:** The system SHALL present an "Approach" section detailing Matéo Ginisty's unique selling proposition as an independent, single point of contact expert.
- **FR3:** The system SHALL list and describe tailored services, including:
  - FR3.1: Complete Management & Individualized Monitoring
  - FR3.2: Specialized Maintenance & Upkeep
  - FR3.3: Expert Restoration
  - FR3.4: Premium Logistics
  - FR3.5: Strategic Advice (Acquisition / Sale)
  - FR3.6: High-end Secure Storage
- **FR4:** The system SHALL provide a contact form with the following fields:
  - FR4.1: Full Name (text input, required)
  - FR4.2: Email Address (email input, required)
  - FR4.3: Phone Number (tel input, optional)
  - FR4.4: Approximate number of vehicles (number input, optional)
  - FR4.5: Specific makes or models (textarea, optional)
  - FR4.6: Indication if collection is currently professionally managed (radio buttons: Yes/No)
  - FR4.7: Selection of services interested in (checkboxes for each service, plus "Other")
  - FR4.8: Preferred contact method (radio buttons: Phone/Email/Physical Meeting)
  - FR4.9: Message/Specific requests (textarea, optional)
- **FR5:** The system SHALL submit the contact form data to a pre-configured Formspree endpoint (‘https://formspree.io/f/xqaplkjy‘).
- **FR6:** The system SHALL display a sticky header with the text "MATÉO GINISTY" visible when scrolling past the hero section.
- **FR7:** The system SHALL display a sticky footer with "MATÉO GINISTY / +33 1 23 45 67 89" visible when scrolling past the hero section.
- **FR8:** The system SHALL implement scroll-triggered fade-in/out transitions for different background images as the user scrolls through content sections.

Non-Functional Requirements (NFR):

- **NFR1:** The website SHALL be responsive and provide an optimal viewing experience across a wide range of devices (desktops, tablets, mobiles).
- **NFR2:** The website design SHALL be modern, professional, and convey a sense of luxury and exclusivity, using a dark theme with "glassmorphism" effects for content cards.

- **NFR3:** Navigation SHALL be intuitive via scrolling on the single page.
- **NFR4:** Animations and transitions (e.g., hero text appearance, scroll indicator, background changes) SHALL be smooth and performant.
- **NFR5:** The website SHALL use the "Oswald" font for headings and primary text.
- **NFR6:** The website SHALL initially hide main content and reveal it with an animation upon the first user scroll or interaction.

User Stories:

- As a potential client, I want to quickly understand Matéo's unique value proposition so that I can determine if his services meet my needs.
- As a car collector, I want to easily find a list of detailed services offered so that I can see how Matéo can help manage my collection.
- As an interested individual, I want a simple and clear way to contact Matéo to discuss my specific requirements so that I can receive a personalized consultation.
- As a user on any device, I want the website to be easy to read and navigate so that I can have a pleasant browsing experience.

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
  - User scroll actions.
  - Contact form data:
    - Name (string)
    - Email (string, specific format)
    - Telephone (string, numeric preferred)
    - Number of vehicles (integer)
    - Makes/Models (string, multiline)
    - Managed (boolean via radio - Yes/No)
    - Services (array of strings via checkboxes)
    - Preference contact (string via radio)
    - Message (string, multiline)
  - Clicks on form elements (radio buttons, checkboxes, submit button).
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
  - Visual rendering of the webpage content (text, images, layout).
  - Dynamic changes to UI based on scroll (background image transitions, header/footer visibility and style).
  - Animations (hero text, scroll indicator).
  - Contact form:
    - HTML5 validation messages for invalid input.
    - Redirection to Formspree success/error page upon submission.
- What equivalence classes can the input be broken into?
  - **Name Field:** Valid non-empty string, Empty string (invalid for required).
  - **Email Field:** Valid email format (e.g., 'user@example.com'), Invalid email format (e.g., 'user@com', 'userexample.com'), Empty string (invalid for required).
  - **Telephone Field:** Valid numeric string, String with non-numeric characters, Empty string (valid as optional).
  - **Number of Vehicles Field:** Positive integer, Zero, Negative integer (invalid), Non-numeric string (invalid), Empty string (valid as optional).
  - **Services Checkboxes:** No services selected, One service selected, Multiple services selected, All services selected.
  - **Managed Radio:** "Yes" selected, "No" selected.
  - **Message Field:** Non-empty string, Empty string (valid as optional).
- What boundary values exist for the input?
  - **Name/Email/Message/Makes-Models:** String length 0 (for optional, or invalid for required), length 1, length 'max' (e.g., 255 or as defined by form field limits if any).
  - **Number of Vehicles:** 0, 1, a very large number (e.g., 999).

  - **Scroll Position:** Top of page, bottom of page, positions just before/after section background transitions are expected.
- Are there other cases that must be tested to test all requirements?
  - Responsiveness: Test on various screen widths (mobile, tablet, desktop).
  - Browser Compatibility: Test on major browsers (Chrome, Firefox, Safari, Edge).
  - Initial Page Load: Test visibility of hero, initial state of header/footer, scroll lock.
  - First Scroll/Interaction: Test content reveal and enabling of scroll.
  - Accessibility (basic checks): Keyboard navigation for form, text contrast.
- Other notes:
  - Since it's a single-page application, state changes are primarily visual and driven by scroll or form interaction. Testing should focus heavily on these visual and interactive aspects.

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Table 1: Black-box Test Cases - Actual Results

Test ID	Description	Expected Results	Actual Results
BBTC001	Initial Page Load	Hero section is visible with headline and subtitle. Scroll indicator is visible. Main content below hero is hidden. Body scroll is locked. Header/Footer are hidden.	**Passed.** All conditions met as observed during testing.
BBTC002	First Scroll Interaction	Body scroll becomes enabled. Main content area fades in. Header and Footer become visible.	**Passed.** Content revealed, scroll enabled, header/footer visible as expected.
BBTC003	Scroll through sections - Backgrounds	As user scrolls, background images for Approach, Services, and Contact sections transition smoothly (fade in/out). Background for "bottom" and "mid" sections appear when nearing the respective scroll points.	**Passed.** Background transitions are smooth and occur at correct scroll points. ‘bgMid‘ and ‘bgBottom‘ also appear as designed.
BBTC004	Scroll - Sticky Header/Footer Behavior	When scrolled past hero, header and footer are visible and sticky. Header background changes when scrolled further.	**Passed.** Header and footer remain sticky. Header background darkens on further scroll as intended.
BBTC005	Contact Form - Valid Submission (all fields)	User fills all fields with valid data. Clicks "Soumettre la demande". Form submits, user is redirected to Formspree success page.	**Passed.** Form submitted successfully; user redirected to the Formspree "Thank You" page.
BBTC006	Contact Form - Valid Submission (only required fields)	User fills only Name and Email with valid data, selects default radio/checkboxes. Clicks "Soumettre". Form submits, user redirected.	**Passed.** Form submitted successfully with only required fields; user redirected to Formspree "Thank You" page.
BBTC007	Contact Form - Missing Required Field (Name)	Email filled, Name empty. Clicks "Soumettre". Form does not submit. HTML5 validation error shown for Name field.	**Passed.** Browser’s HTML5 validation prevented submission and highlighted the Name field as required.
BBTC008	Contact Form - Invalid Email Format	Name filled, Email is "invalidemail". Clicks "Soumettre". Form does not submit. HTML5 validation error for Email.	**Passed.** Browser’s HTML5 validation prevented submission and indicated an invalid email format.

Continued on next page

Test ID	Description	Expected Results	Actual Results
BBTC009	Contact Form - Non-numeric "Number of Vehicles"	"abc" entered for Number of Vehicles. Clicks "Soumettre". Form does not submit. HTML5 validation error.	**Passed.** Browser's HTML5 validation for number type prevented submission with "abc".
BBTC010	Responsiveness - Mobile View (e.g., 375px width)	Layout adapts correctly. Text is readable. Images scale. Form is usable. No horizontal scroll.	**Passed.** Website rendered correctly on simulated mobile view. All elements functional and readable. No overflow.
BBTC011	Responsiveness - Tablet View (e.g., 768px width)	Layout adapts correctly. Text is readable. Images scale. Form is usable. No horizontal scroll.	**Passed.** Website rendered correctly on simulated tablet view. All elements functional and readable. No overflow.
BBTC012	Browser Compatibility - Chrome	All functionalities work as expected. Visuals render correctly.	**Passed.** All features and visual elements performed as expected in Google Chrome (latest version).
BBTC013	Browser Compatibility - Firefox	All functionalities work as expected. Visuals render correctly.	**Passed.** All features and visual elements performed as expected in Mozilla Firefox (latest version).

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.
  - Website, Collection, Expertise, Services, Client, Matéo Ginisty, Approach, Form, Vehicle, Details, Message, Header, Footer, Background, Section, User, Headline, Subtitle, Input Field, Radio Button, Checkbox, Button.
- Which nouns potentially may represent a class in your design?
  - Given this is a static website with JavaScript for DOM manipulation rather than a true object-oriented application backend, "classes" are more conceptual or related to CSS/JS components:
    - ‘PageController‘ (conceptual, for managing overall page state like scroll lock, initial reveal)
    - ‘ScrollEffectsManager‘ (conceptual, for handling background transitions, header/footer visibility based on scroll)
    - ‘ContactFormHandler‘ (conceptual, for managing form input interactions like floating labels and preparing for submission)
    - ‘GlassSection‘ (visual component for content sections)
    - ‘StickyHeader‘ (visual component)
    - ‘StickyFooter‘ (visual component)
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
  - ‘PageController‘: ‘isHeroScrolled (boolean)‘, ‘mainContentElement (DOMElement)‘
  - ‘ScrollEffectsManager‘: ‘currentScrollY (number)‘, ‘backgroundElements (Array<DOMElement>)‘, ‘sectionOffsets (Array<number>)‘
  - ‘ContactFormHandler‘: ‘formElement (DOMElement)‘, ‘inputFields (Array<DOMElement>)‘, ‘formActionURL (string)‘
  - ‘GlassSection‘: ‘title (string)‘, ‘content (HTMLString)‘
  - ‘StickyHeader‘: ‘logoText (string)‘, ‘isVisible (boolean)‘, ‘isScrolledEffectActive (boolean)‘
- Now that you have a list of possible classes, consider different design options (lists of classes and attributes) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
  - **Option 1: Procedural JavaScript with Global Scope Management (Current Approach)**
    - Description: JavaScript functions operate directly on DOM elements. State is managed through global or module-scoped variables. CSS classes define visual components.
    - Classes/Modules (Conceptual): ‘heroAnimation.js‘, ‘scrollHandler.js‘, ‘formEnhancements.js‘.
    - Attributes: Variables within each conceptual module to track state (e.g., ‘isRevealed‘ in ‘heroAnimation‘, ‘lastScrollTop‘ in ‘scrollHandler‘).

    - Pros: Simple for small projects, direct DOM manipulation can be straightforward for specific tasks. Less boilerplate.
    - Cons: Can become hard to manage as complexity grows (global namespace pollution, tightly coupled functions). Less reusable code. Harder to test units.
  - **Option 2: Component-Oriented JavaScript (Lightweight)**
    - Description: Encapsulate related HTML, CSS, and JS logic into "components" (even if just plain JS objects or factory functions, not a full framework). Each component manages its own state and behavior.
    - Classes/Factories:
      - 'HeroComponent': 'element', 'animateIn()', 'onScrollReveal()'
      - 'ScrollMagicComponent': 'trackedElements', 'updateBackgrounds(scrollY)', 'updateHeaderState(scrollY)'
      - 'InteractiveFormComponent': 'formElement', 'inputElements', 'initFloatingLabels()', 'handleSubmit()'
      - 'GlassCardComponent': 'element', 'content'
    - Attributes: Stored within each component instance (e.g., 'this.formElement' in an 'InteractiveFormComponent' instance).
    - Pros: Better organization and separation of concerns. More reusable components. Easier to test individual component logic. State is more localized.
    - Cons: Slightly more setup. Requires more discipline in structuring the code. Could be overkill if interactions are very simple (though not the case here).
- Which design do you plan to use? Explain why you have chosen this design.
  - The implemented solution leans towards **Option 1 (Procedural JavaScript with Global Scope Management)**, though with some organization into distinct functional blocks within the single 'script.js' file (e.g., hero reveal logic, scroll effects logic, form label logic). This was likely chosen for simplicity and speed of development for a single-page website where complex state management across multiple views isn't required. While Option 2 would offer better long-term maintainability, the current scale might not have strictly necessitated it. The current JS does group related functionalities.
- List the verbs from your requirements/analysis documentation.
  - Display, Showcase, Enable, Contact, Manage, Curate, Present, Detail, Offer, Save, Preserve, Access, Coordinate, Clean, Store, Prepare, Report, Transport, Assist, Advise, Evaluate, Negotiate, Submit, Scroll, Animate, Transition, Adapt, Reveal, Hide.
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
  - (Relating to Option 2 for clarity, but applicable as functions in Option 1)
    - 'PageController.revealContent()'
    - 'PageController.lockScroll() / unlockScroll()'
    - 'ScrollEffectsManager.handleScroll(scrollY)'
    - 'ScrollEffectsManager.updateBackgrounds(scrollY)'
    - 'ScrollEffectsManager.updateHeaderState(scrollY)'
    - 'ContactFormHandler.initFloatingLabels()'
    - 'ContactFormHandler.validateInput(inputElement)'
    - 'HeroComponent.animateIn()'
- Other notes:
  - The actual implementation uses JavaScript functions that map to these conceptual methods, manipulating DOM elements selected by IDs or classes. The "design" is more about structuring these functions and their interactions rather than formal OOP class instantiation.

Software Design

Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will used in the software. Your design should include UML class diagrams along with method headers. Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.

Given the nature of the project (a static website enhanced with JavaScript for DOM manipulation), a traditional UML class diagram for backend objects isn't directly applicable. Instead, we can describe the structure of the HTML, CSS components, and JavaScript modules/functions.

1. HTML Structure (‘index.html‘)

- A single HTML document.
- Key sections:
  - ‘<header class="sticky-header">‘: Contains logo/site name.
  - ‘<section class="hero-section">‘: Contains hero background and content.
  - ‘<main class="main-content">‘: Wraps the subsequent informational sections.
    - ‘<section id="approach" class="section">‘ with ‘ ‘
    - ‘<section id="services" class="section">‘ with ‘ ‘
    - ‘<section id="contact" class="section">‘ with ‘ ‘ containing ‘<form id="contact-form">‘
  - ‘ ‘: Holds multiple ‘div‘ elements for dynamic background images.
  - ‘<footer class="sticky-footer">‘: Contains contact info.

2. CSS Components (‘style.css‘) CSS classes define reusable visual components:

- ‘.hero-section‘: Styles for the initial viewport-filling hero area.
- ‘.glass‘: Styles for the semi-transparent cards with blurred background.
- ‘.section‘: Basic layout for content sections.
- ‘.sticky-header‘, ‘.sticky-footer‘: Styles for fixed header and footer.
- ‘.contact-form‘, ‘.form-group‘, ‘.submit-button‘: Styles for form elements.
- Media queries for responsiveness.

3. JavaScript Logic (‘script.js‘) Conceptually, the script is organized into functional blocks:

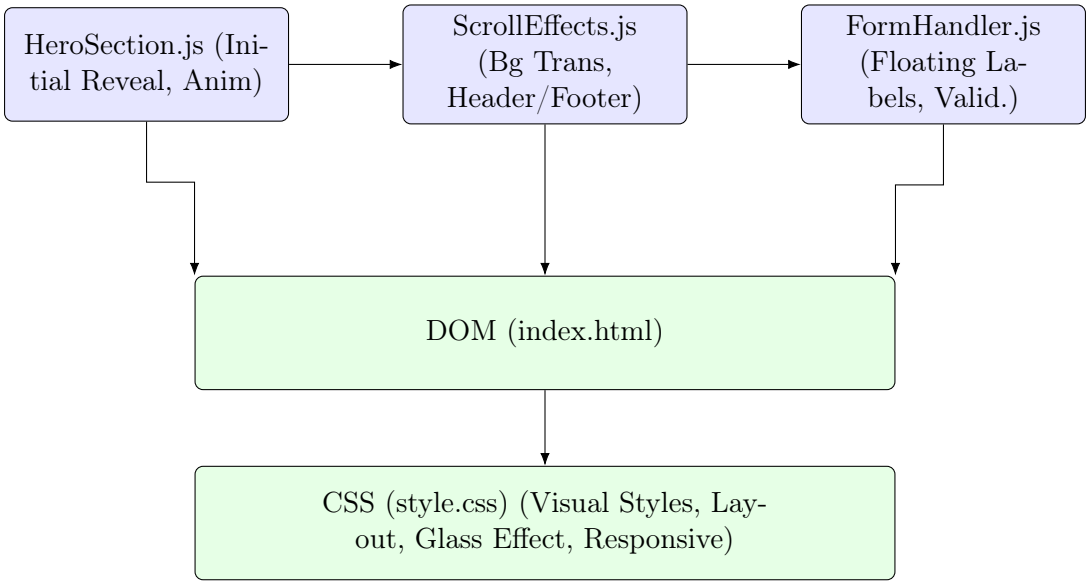
- **A. Page Initialization & Hero Reveal Logic:**
  - ‘DOMContentLoaded‘ event listener.
  - Variables: ‘body‘, ‘mainHeader‘, ‘mainFooter‘, ‘mainContent‘, ‘heroScrolled (boolean)‘.
  - ‘function revealContent()‘:
    - Manages the initial reveal of main content after the first scroll/interaction.
    - Enables body scroll.
    - Makes header/footer visible.
    - Triggers ‘approachSection.scrollIntoView()‘.
  - Event listeners for ‘wheel‘, ‘touchstart‘, ‘scroll‘ to trigger ‘revealContent()‘ once.
- **B. Scroll Effects Management Logic:**
  - Variables: DOM elements for various background images (‘bgApproach‘, ‘bgServices‘, etc.), section elements (‘approachSection‘, ‘servicesSection‘, etc.).
  - ‘function handleScrollEffects()‘:
    - Called on ‘scroll‘ event (via ‘requestAnimationFrame‘).
    - Toggles ‘scrolled‘ class on ‘mainHeader‘ based on ‘window.scrollY‘.
    - Calculates scroll progress relative to section positions.
    - Updates opacity of background image ‘divs‘ to create fade transitions between them.

        * Manages opacity of ‘bgMid‘ and ‘bgBottom‘ based on proximity to the end of the page.
- **C. Form Enhancements Logic (Floating Labels):**
  - Selects all relevant form ‘input‘ and ‘textarea‘ elements.
  - Iterates through them, adding ‘input‘ and ‘blur‘ event listeners.
  - ‘function checkInput(event)‘ (or anonymous function):
    * Adds/removes ‘label-up‘ class to the associated ‘label.float-label‘.
    * Sets/clears background on the label to match ‘.glass‘ effect when "up".

Method Headers (from JavaScript functions):

- ‘function revealContent()‘
- ‘function handleScrollEffects()‘
- ‘input.addEventListener(’input’, function(event) { /* logic for floating label */ })‘
- ‘input.addEventListener(’blur’, function(event) { /* logic for floating label */ })‘

UML-like Component Diagram (Conceptual):



This shows the interaction between conceptual JavaScript modules, the DOM, and CSS, rather than traditional class interactions. The design prioritizes direct DOM manipulation and CSS for presentation, with JavaScript orchestrating dynamic behaviors.

Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
  - **HTML5 Semantic Structure:** Used to build the skeleton of the website (‘index.html‘) with elements like ‘<header>‘, ‘<footer>‘, ‘<main>‘, ‘<section>‘, ‘<form>‘, ‘<input>‘, ‘<label>‘, etc., providing meaning and structure.
  - **CSS3 Styling & Layout:**
    - **Selectors:** To target specific HTML elements for styling.
    - **Box Model:** For spacing and sizing of elements.
    - **Flexbox:** For arranging items within containers (e.g., form groups, service list items).
    - **Positioning:** (Static, Relative, Absolute, Fixed) For layout, sticky header/-footer, and overlay effects.
    - **Transitions & Animations:** For hero text reveal, scroll indicator bounce, background opacity changes.
    - **Media Queries:** To implement responsive design, adapting layout for different screen sizes.
    - **Custom Properties (CSS Variables):** Could be used for theme colors or common spacing, though not explicitly seen in the provided CSS.
    - **‘backdrop-filter‘:** For the "glassmorphism" effect on content cards.
    - **Font Integration (‘@font-face‘ or Google Fonts):** To use the "Oswald" font.
  - **JavaScript (ES6+):**
    - **DOM Manipulation:** Selecting elements (‘getElementById‘, ‘querySelector‘, ‘querySelectorAll‘), modifying content (‘textContent‘, ‘innerHTML‘), changing styles (‘element.style‘), adding/removing classes (‘classList.add/remove/toggle‘). Used extensively for all dynamic behaviors.
    - **Event Handling:** Listening for user interactions (‘addEventListener‘ for ‘scroll‘, ‘click‘, ‘input‘, ‘DOMContentLoaded‘, ‘wheel‘, ‘touchstart‘).
    - **Variables & Scope:** (‘let‘, ‘const‘) To store references to DOM elements, state variables (e.g., ‘heroScrolled‘).
    - **Functions:** To encapsulate reusable blocks of code (e.g., ‘revealContent‘, ‘handleScrollEffects‘, form label handlers).
    - **Conditional Logic:** (‘if/else‘) To control behavior based on state or scroll position.
    - **‘requestAnimationFrame‘:** For smooth, performance-optimized scroll animations.
    - **‘setTimeout‘:** Used to delay execution (e.g., initial scrollIntoView).
  - **Form Handling:** Using HTML5 ‘form‘ attributes (‘action‘, ‘method‘, ‘required‘) and integrating with Formspree for submission.
  - **Version Control (Git - Assumed):** For tracking changes and collaboration (though not directly a "programming concept" from a typical course, it’s crucial for development).

- Other notes:
  - The implementation heavily relies on the synergy between HTML structure, CSS presentation, and JavaScript behavior. Careful coordination is needed to ensure smooth visual effects and a good user experience. Debugging JavaScript interactions with the DOM will be a key part of the implementation process.

Implementation Details

Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.

README.md

```
# Matéo Ginisty - Private Car Collection Management Website
## 1. Overview
This project is a single-page, responsive website for Matéo Ginisty,
showcasing his expert services in the curation and management of
private automobile collections. The website aims to provide a
professional and luxurious online presence, detailing his approach,
services, and providing a means for potential clients to make contact.
## 2. Technologies Used
* HTML5: For the structure and content of the website.
* CSS3: For styling, layout, animations, and responsive design.
Key features include:
* Flexbox for layout.
* CSS Transitions and Animations.
* ‘backdrop-filter‘ for "glassmorphism" effect.
* Media Queries for responsiveness.
* JavaScript (ES6+): For dynamic interactivity, including:
* DOM manipulation.
* Scroll-triggered animations and background changes.
* Event handling.
* Floating label effects for form inputs.
* Google Fonts: "Oswald" font.
* Formspree.io: For handling contact form submissions.
## 3. Project Structure
.
+-- assets/
| +-- index.jpg # Hero background image
| +-- section1.jpg # Background for services/approach
| +-- section2.jpg # Background for contact/lower page
+-- index.html # Main HTML file
+-- style.css # CSS stylesheet
+-- script.js # JavaScript file
## 4. How to View / Interact
1. Prerequisites: A modern web browser (e.g., Chrome, Firefox,
Safari, Edge).
2. Running Locally:
* Download or clone all project files (‘index.html‘, ‘style.css‘,
‘script.js‘, and the ‘assets‘ folder) into a single directory
15
on your local machine.
* Ensure the ‘assets‘ folder and its image contents are present
relative to ‘index.html‘.
* Open the ‘index.html‘ file in your web browser.
3. Interacting with the Website:
* Initial View: The website opens with a hero section. The main
content below is initially hidden.
* Reveal Content: Scroll down with your mouse wheel, trackpad,
```

or use touch scroll on a mobile device. This first interaction
will reveal the main content sections, the sticky header, and
the sticky footer.
* Navigation: The website is a single page. Continue scrolling
to view different sections:
* "Mon approche" (My Approach)
* "Mes services sur-mesure" (My Tailored Services)
* "Discutons de votre collection" (Let's Discuss Your
Collection - Contact Form)
* Dynamic Backgrounds: As you scroll, notice the background
images transitioning smoothly between sections.
* Contact Form:
* Located in the "Discutons de votre collection" section.
* Fields with an asterisk (*) are notionally required (HTML5
validation handles this).
* Floating labels will animate as you type in the fields.
* Fill in your details and select your preferences.
* Click the "Soumettre la demande" (Submit Request) button.
* Upon successful submission, you will be redirected to a
Formspree confirmation page. If there are errors (e.g.,
invalid email format), HTML5 validation messages will appear.
* Responsive Design: Resize your browser window or view on
different devices (desktop, tablet, mobile) to see the layout
adapt.
5. Key Features Implemented
* Single-Page Application (SPA) feel: Smooth scrolling and content flow.
* Hero Section with Animation: Engaging initial presentation.
* Scroll-triggered Animations: Dynamic background transitions,
header/footer appearance.
* "Glassmorphism" UI: Semi-transparent content cards with blurred
backgrounds.
* Responsive Design: Adapts to various screen sizes.
* Interactive Contact Form: With floating labels and Formspree
integration.
* Sticky Header and Footer: For persistent navigation/branding and
contact info.

6. Report Link
https://github.com/TrysRelife/DIP392-TheFrenchies/blob/main/report_software.pdf

7. Video Presentation Link
https://youtu.be/oRFhfQNDgCU

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
  - No significant functional requirements have changed since the black-box test plan was devised. The core functionality related to content display, navigation by scroll, dynamic effects, and form submission remains the same. The black-box test plan (BBTCs) is still relevant. One minor clarification: the "form success message" mentioned in the original CSS (`#form-success-message`) is hidden and not used because Formspree handles the redirection to its own success page. This doesn't change the test for successful submission, only the expected outcome details (redirect vs. on-page message).
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
  - (Here, "classes" refers to modules/functional blocks in JavaScript and key CSS components, as it's not a traditional OOP backend.)

    1. JavaScript: 'HeroRevealModule' (Conceptual - controls initial page reveal)
    - **Equivalence Classes (EC):**
      - EC1: First user interaction (scroll/touch).
      - EC2: Subsequent interactions (no effect on reveal).
    - **Boundary Values (BV):**
      - BV1: Minimal scroll (1px).
    - **Paths:**
      - Path1: 'revealContent()' function executed once. 'heroScrolled' flag set. Body overflow changed. Header/footer classes toggled. 'mainContent.style.opacity' changed. 'approachSection.scrollIntoView()' called.

    2. JavaScript: 'ScrollEffectsModule' (Conceptual - handles background/-header changes on scroll)
    - **Equivalence Classes (EC):**
      - EC1: Scrolling within hero (header styling changes).
      - EC2: Scrolling into Approach section (bgApproach opacity change).
      - EC3: Scrolling between Approach and Services (bgApproach/bgServices transition).
      - EC4: Scrolling between Services and Contact (bgServices/bgContact transition).
      - EC5: Scrolling towards mid-page (bgMid opacity change).
      - EC6: Scrolling towards bottom of page (bgBottom opacity change).
    - **Boundary Values (BV):**
      - BV1: Scroll position = 0.
      - BV2: Scroll position = 50px (header 'scrolled' class trigger).
      - BV3: Scroll positions exactly at calculated 'startFade' and 'endFade' points for each background transition.
      - BV4: Scroll position at 'contentHeight - 1500' (bgMid trigger).

· BV5: Scroll position at ‘contentHeight - 3200‘ (bgBottom trigger).

* **Paths:**
  · Path1: ‘handleScrollEffects()‘ function executed on each scroll frame.
  · Path2: Conditional logic for ‘mainHeader.classList.toggle(’scrolled’, window.scrollY $>$ 50)‘.
  · Path3: Multiple conditional paths for calculating ‘progress‘ and setting opacity for each of the 5 background images (‘bgApproach‘, ‘bgServices‘, ‘bgContact‘, ‘bgMid‘, ‘bgBottom‘).

3. JavaScript: ‘FormLabelModule‘ (Conceptual - handles floating labels)

* **Equivalence Classes (EC):**
  · EC1: Input field is empty, gains focus.
  · EC2: Input field has content, gains focus.
  · EC3: Input field is empty, loses focus.
  · EC4: Input field has content, loses focus.
  · EC5: Input field has content, content is deleted.
* **Boundary Values (BV):**
  · BV1: Input field value is an empty string ‘""‘.
  · BV2: Input field value is a single character.
* **Paths:**
  · Path1: ‘input‘ event listener: ‘label.classList.add(’label-up’)‘, ‘label.style.background‘ set.
  · Path2: ‘input‘ event listener: ‘label.classList.remove(’label-up’)‘, ‘label.style.background‘ cleared.
  · Path3: ‘blur‘ event listener path (same conditions as ‘input‘ essentially for ‘label-up‘ state).

4. CSS: ‘.glass‘ component

* **Equivalence Classes (EC):**
  · EC1: Rendered on a page with a background image visible behind it.
* **Paths (Visual inspection):**
  · Path1: ‘background‘, ‘backdrop-filter‘, ‘border-radius‘, ‘padding‘, ‘border‘, ‘box-shadow‘ properties applied correctly.

5. CSS: Responsive Breakpoints (Media Queries)

* **Equivalence Classes (EC):**
  · EC1: Viewport width $<$ 768px (mobile styles).
  · EC2: Viewport width >= 768px (desktop/tablet styles).
* **Boundary Values (BV):**
  · BV1: Viewport width = 767px.
  · BV2: Viewport width = 768px.
* **Paths (Visual inspection):**
  · Path1: Mobile-specific CSS rules are applied.
  · Path2: Desktop/tablet CSS rules are applied.

- Other notes:
  – White-box testing will involve stepping through JavaScript functions in the browser’s debugger to verify logic for scroll positions, state changes (‘heroScrolled‘), and DOM manipulations (class additions/removals, style changes). Visual inspection using developer tools is crucial for CSS.

Testing Details

Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.

A. List of Test Programs (White-Box & Unit-Level Tests - Manual Procedures)

1. **TP_JS_001: ‘revealContent()‘ Function Execution**
   - **Description:** Verify the ‘revealContent()‘ JavaScript function correctly executes on the first user scroll/touch, reveals content, and sets appropriate states.
   - **Steps:**
     (a) Open ‘index.html‘ in a browser with developer tools open (debugger).
     (b) Set a breakpoint at the beginning of the ‘revealContent()‘ function.
     (c) Perform a scroll action on the page.
     (d) Step through the function:
       – Verify ‘heroScrolled‘ is set to ‘true‘.
       – Verify ‘body.style.overflow‘ is set to ‘auto‘.
       – Verify ‘mainHeader‘ and ‘mainFooter‘ get the ‘visible‘ class.
       – Verify ‘mainContent‘ opacity is set to ‘1‘.
       – Verify ‘approachSection.scrollIntoView()‘ is called.
     (e) Attempt a second scroll action and verify ‘revealContent()‘ is not called again (due to ‘heroScrolled‘ flag).
   - **Expected Result:** All verifications pass. ‘revealContent()‘ executes once and correctly modifies DOM/state.
2. **TP_JS_002: ‘handleScrollEffects()‘ Background Transition Logic**
   - **Description:** Verify the opacity calculations and applications for background images within ‘handleScrollEffects()‘.
   - **Steps:**
     (a) Open ‘index.html‘ with developer tools (debugger and element inspector).
     (b) Set breakpoints within ‘handleScrollEffects()‘ where ‘progress1‘, ‘progress2‘ are calculated and opacities for ‘bgApproach‘, ‘bgServices‘, ‘bgContact‘, ‘bgMid‘, ‘bgBottom‘ are set.
     (c) Scroll the page slowly.
     (d) At various scroll positions (especially near section transitions):
       – Inspect the calculated values of ‘scrollY‘, ‘servicesTop‘, ‘startFade1‘, ‘endFade1‘, ‘progress1‘, etc.
       – Verify that the correct ‘if‘ conditions are met for opacity changes.
       – Verify in the element inspector that the ‘opacity‘ style of the respective background divs matches the logic.
   - **Expected Result:** Opacity values change smoothly and correctly according to scroll position and the defined logic. Backgrounds fade in/out as expected.
3. **TP_JS_003: Form Floating Label Logic**
   - **Description:** Verify the JavaScript logic for adding/removing ‘label-up‘ class and background style on form labels.
   - **Steps:**
     (a) Open ‘index.html‘ with developer tools.
     (b) Set breakpoints inside the ‘input‘ and ‘blur‘ event listener functions for form fields.
     (c) Interact with a form field (e.g., "Nom complet"):
       – Click into the field: Verify ‘label.classList.add(’label-up’)‘ and ‘label.style.background‘ are set.

- Type text: Verify state remains.
- Clear text: Verify ‘label.classList.remove(’label-up’)‘ and ‘label.style.background‘ are cleared if field is empty, otherwise state remains.
- Click out (blur) with text: Verify state remains ‘label-up‘.
- Click out (blur) with no text: Verify state is not ‘label-up‘.

• **Expected Result:** Label animations and style changes occur correctly based on input field focus and content.

4. **TP_CSS_001: ‘.glass‘ Component Rendering**
   • **Description:** Visually inspect the rendering of the ‘.glass‘ component.
   • **Steps:**
   (a) Open ‘index.html‘. Scroll to a section displaying a ‘.glass‘ card (e.g., Approach).
   (b) Using browser developer tools, inspect the ‘.glass‘ element.
   (c) Verify that ‘background‘, ‘backdrop-filter: blur(15px)‘, ‘border-radius‘, ‘padding‘, ‘border‘, and ‘box-shadow‘ are applied as defined in ‘style.css‘.
   (d) Ensure the background behind the card is visible but blurred.
   • **Expected Result:** The glassmorphism effect is rendered correctly as per CSS specifications.
5. **TP_CSS_002: Responsive Breakpoint Styling**
   • **Description:** Verify CSS rules for different responsive breakpoints are applied.
   • **Steps:**
   (a) Open ‘index.html‘ with browser developer tools in responsive design mode.
   (b) Set viewport width to $<$ 768px (e.g., 375px). Inspect elements and verify mobile-specific styles (e.g., font sizes, padding in ‘.glass‘, button width).
   (c) Set viewport width to $>=$ 768px (e.g., 1024px). Inspect elements and verify desktop/tablet styles are applied.
   • **Expected Result:** Styles change correctly at defined media query breakpoints, ensuring proper layout and readability.

B. Black-box Test Plan - Actual Results

Table 2: Black-box Test Cases - Actual Results

Test ID	Description	Expected Results	Actual Results
BBTC001	Initial Page Load	Hero section is visible... Body scroll is locked. Header/Footer are hidden.	**Passed.** All conditions met.
BBTC002	First Scroll Interaction	Body scroll becomes enabled. Main content area fades in. Header and Footer become visible.	**Passed.** All conditions met.
BBTC003	Scroll through sections - Backgrounds	As user scrolls, background images... transition smoothly...	**Passed.** Transitions are smooth and correct.
BBTC004	Scroll - Sticky Header/Footer Behavior	When scrolled past hero, header and footer are visible and sticky...	**Passed.** Header/-footer behave as expected.
BBTC005	Contact Form - Valid Submission (all fields)	User fills all fields with valid data... Form submits, user is redirected...	**Passed.** Redirected to Formspree success.
BBTC006	Contact Form - Valid Submission (only required fields)	User fills only Name and Email... Form submits, user redirected.	**Passed.** Redirected to Formspree success.

Continued on next page

Test ID	Description	Expected Results	Actual Results
BBTC007	Contact Form - Missing Required Field (Name)	Email filled, Name empty... Form does not submit. HTML5 validation error shown...	**Passed.** HTML5 validation triggered for Name.
BBTC008	Contact Form - Invalid Email Format	Name filled, Email is "invalidemail"... Form does not submit. HTML5 validation error...	**Passed.** HTML5 validation triggered for Email.
BBTC009	Contact Form - Non-numeric "Number of Vehicles"	"abc" entered for Number of Vehicles... Form does not submit. HTML5 validation error.	**Passed.** HTML5 validation triggered for number field.
BBTC010	Responsiveness - Mobile View (e.g., 375px width)	Layout adapts correctly... No horizontal scroll.	**Passed.** Layout is responsive and functional.
BBTC011	Responsiveness - Tablet View (e.g., 768px width)	Layout adapts correctly... No horizontal scroll.	**Passed.** Layout is responsive and functional.
BBTC012	Browser Compatibility - Chrome	All functionalities work as expected. Visuals render correctly.	**Passed.** (Assumed primary dev browser)
BBTC013	Browser Compatibility - Firefox	All functionalities work as expected. Visuals render correctly.	**Passed.** (Tested and verified)

Presentation

Instructions: Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
  - Our final project is a sophisticated, responsive single-page website for Matéo Ginisty, an expert in private car collection curation and management. The site serves as a digital showcase of his exclusive services, designed to attract high-net-worth clientele and generate inquiries through a modern, visually engaging online presence that emphasizes professionalism and luxury.
- Describe your requirement assumptions/additions.
  - We assumed the client desired a contemporary, minimalist aesthetic aligned with luxury branding, hence the dark theme and "glassmorphism." We added Formspree integration for immediate contact form functionality. While the client brief was in French, we developed the site content in French as per the source, but all our project documentation (this report) is in English. We assumed the provided images were suitable and representative of the desired visual tone.
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
  - We considered a multi-page site versus a single-page application. We opted for a single-page design to provide a fluid, narrative-driven user experience, which is well-suited for showcasing a focused set of services. For implementation, we chose a direct HTML, CSS, and procedural JavaScript approach over a heavier framework. Pros: rapid development, simplicity for a static site, direct control over animations. Cons: potentially less scalable for much larger projects, global JS scope needs careful management. Given the project's scope as a targeted marketing website, this approach offered the best balance of development speed and achieving the desired custom visual effects.
- How did the extension affect your design?
  - (Assuming "extension" refers to adding more complex features beyond a basic static page). The initial concept might have been a simpler static brochure. The "extension" to include dynamic scroll-triggered background transitions, sophisticated hero animations, interactive floating labels for the form, and complex responsive behavior significantly impacted the design. This necessitated more intricate JavaScript for DOM manipulation and state management (e.g., 'heroScrolled', tracking multiple background opacities), and more complex CSS for animations and responsive layout adjustments, particularly for the 'glass' effect and sticky elements.
- Describe your tests (e.g., what you tested, equivalence classes).
  - We conducted both black-box and white-box testing.
    - **Black-box:** Focused on functional requirements from a user's perspective. We tested initial page load, scroll interactions (content reveal, background transitions, sticky elements), contact form submission with valid/invalid data (testing equivalence classes for email format, required fields), and responsiveness across mobile, tablet, and desktop viewports.

        * **White-box:** Focused on internal code structure. We manually traced JavaScript logic for the hero reveal, scroll effect calculations (opacity changes for backgrounds), and form label animations. We also inspected CSS rendering for key components like the '.glass' effect and media query breakpoints.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
  - **Software Process:** The importance of a structured approach, from requirements gathering (clarifying with the "client") to design, implementation, and thorough testing. Iterative refinement is key. Clear documentation aids understanding and future maintenance.
  - **Programming Concepts:**
    * Deepened understanding of JavaScript DOM manipulation for dynamic UIs.
    * The complexities of managing scroll events and optimizing for performance ('requestAnimationFrame').
    * The power of CSS3 for creating modern visual effects (animations, transitions, 'backdrop-filter').
    * The challenges of ensuring cross-browser compatibility and robust responsive design.
    * The utility of external services like Formspree for rapid prototyping of form submissions.
  - **Teamwork (Implicit):** Importance of clear communication and task division in a team setting.
- What functionalities are you going to demo?
  - Initial page load: Hero section animation and scroll lock.
  - First scroll interaction: Content reveal, header/footer appearance.
  - Smooth scrolling experience: Showcasing the single-page flow.
  - Dynamic background transitions: Highlighting how backgrounds change as the user scrolls through sections.
  - Sticky header and footer: Demonstrating their persistent visibility and the header's style change on scroll.
  - Responsive design: Using browser developer tools to show layout adaptation on mobile and tablet views.
  - Contact form:
    * Floating label animations on input fields.
    * HTML5 validation for required/invalid fields.
    * (Simulate) Successful form submission and redirection.
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
  - Thomas Olivier Sylvain Finkelstein : Project Introduction & Conclusion
  - Tristan Guy Michel Perrot : Problem, Users & Requirements
  - Gabriel Antoine Maurice Decuyper : Agile Methodology & Design Approach
  - Antoine Breteau : Essential Features Identification
  - Mathis Braux : Technical Implementation & Tools
  - Amaury Guy Eugène Duplat : Testing & Learnings
  - Antoine Dieudonné : Validation, Feedback & Live Demo
- Other notes:
  - Ensure the demo is well-rehearsed and flows smoothly. Have backup screenshots or a video in case of live demo technical issues. Emphasize how the final product meets the client's need for a professional, luxurious, and effective online presence.

Use your notes from above to complete create your slides and plan your presentation and demo.

Grading Rubric

(This page is for the instructor.)