Software Requirements Specification (SRS)

System: “The Internet” – UI Component and Web Behavior Demonstration Site
Base URL: **https://the-internet.herokuapp.com/**

---

1. Introduction

1.1 Purpose

The purpose of this SRS is to define the observable functional and non-functional requirements of the public web application hosted at https://the-internet.herokuapp.com/. The site provides a collection of independent examples demonstrating common web UI components, behaviors, and edge cases, primarily for use in web automation, QA training, and exploratory testing scenarios.
The document is based exclusively on behaviors, content, and interactions observable from the deployed site and does not rely on any unverified implementation assumptions.

1.2 Intended Users

The primary intended users inferred from the site content are:

- QA engineers and test automation engineers exploring UI automation scenarios (e.g., dynamic content, frames, JavaScript alerts, broken images, etc.).
- Developers and SDETs (Software Development Engineers in Test) experimenting with locators and asynchronous behaviors.
- Trainers, educators, and learners using the site as a demo system for practicing and demonstrating web testing techniques.

End-users are assumed to be technically proficient and familiar with basic web navigation.

1.3 Scope

The system consists of:

- A home page listing multiple independent “example” pages and links to them.
- A set of self-contained example pages, each demonstrating a specific UI pattern or behavior such as:
    - Static and dynamic content
    - Forms and authentication
    - Alerts and modals
    - Tables and data structures
    - Layout constructs (frames, shadow DOM, large DOM, shifting content)
    - Browser-related behaviors (status codes, multiple windows, context menus, JavaScript errors, geolocation)

Each example page is functionally independent and can be accessed directly via URL or through the homepage navigation.
The scope of this SRS includes:

- All pages and features listed on the home page’s “Available Examples” section.

- Core observable behaviors of each feature page, including UI elements, user-triggered actions, and resulting feedback.
- Cross-cutting behaviors such as navigation and layout.

The scope excludes any back-end systems not visible from the browser and any undocumented behaviors not observable from the UI.

---

2. System Overview

2.1 High-Level Description

"The Internet" application is a static, example-driven site composed of multiple independent demonstration modules. Each module:

- Is accessible via a dedicated URL path, typically /feature_name (e.g., /login, /checkboxes, /dynamic_controls).
- Presents descriptive text explaining the example's purpose, often with explicit statements about how content changes across page loads or interactions (e.g., dynamic content, typos, shifting content).
- Provides minimalistic UI controls (links, buttons, inputs, sliders, tables) with intentionally simple layout and styling.

The site does not expose any global login, profile, or shared session features; the "secure area" is restricted to the context of the login example only.

2.2 Navigation Structure

2.2.1 Home Page

- The root URL (/) displays:
  - A main heading "Welcome to the-internet".
  - A section "Available Examples" listing all example titles as individual links.
- Each list item's text corresponds to the feature name (e.g., "Checkboxes", "Form Authentication", "Broken Images", "Dynamic Content", "Sortable Data Tables", etc.) and links to the respective example page.

2.2.2 Example Pages

- Each example is hosted under a unique path, for example:
  - /checkboxes – checkboxes example
  - /login – form authentication / login page
  - /dropdown – dropdown selection example
  - /dynamic_controls – asynchronous element changes
  - /dynamic_loading – dynamically loaded elements
  - /upload – file uploader
  - /javascript_alerts – JS alert/confirm/prompt examples
  - /drag_and_drop – drag-and-drop example
  - /tables – sortable data tables
  - /notification_message_rendered – random notification messages
  - /entry_ad – entry advertisement modal

- /typos – random typo example
- /add_remove_elements/ – add/remove elements example
- /disappearing_elements – disappearing menu elements
- /hovers – hover behavior with user profiles
- /abtest – A/B test concept description
- /dynamic_content – dynamic text and images
- /status_codes – HTTP status codes information
- /inputs – numeric input field
- /horizontal_slider – slider and displayed value
- /context_menu – right-click context menu alert
- /challenging_dom – complex DOM locators and table
- /exit_intent – exit-intent modal
- /jqueryui/menu – JQuery UI menu example
- /javascript_error – page with JavaScript error on load
- /windows – opening a new window
- /large – large and deep DOM example
- /download – file downloader
- /forgot_password – email-only password retrieval form
- /geolocation – “Where am I?” location button
- /floating_menu – floating menu with long scrollable content
- /infinite_scroll – infinite scroll example
- /frames – entry page to frames examples; links to nested frames and iframe
- /shadowdom – shadow DOM example

(Other linked examples listed on the home page but not fully inspected follow similar patterns and are assumed to be independent demonstration modules, without additional cross-module dependencies.)
Each example usually includes a “Powered by Elemental Selenium” footer line.

2.3 Assumptions and Constraints

2.3.1 Assumptions

- Users access the site via a modern desktop browser capable of executing JavaScript, handling alerts, and supporting HTML5 features (e.g., file uploads, geolocation APIs).
- Users may directly navigate to any example page without visiting the home page first.
- Content and behaviors described on each page (e.g., “typo being introduced randomly” or “content loads new text and images on each refresh”) reflect the intended run-time behavior.

2.3.2 Constraints

- The site is read-only with respect to persistent data. No persistent user data management (registration, database-backed profiles) is exposed.

- Authentication behavior is limited to the “Form Authentication” example context (login page and secure area) and does not extend globally.
- Some behaviors are deliberately non-deterministic or randomized (e.g., dynamic content, random typos, notification message variation), constraining deterministic expectations for exact text on every page load.
- Some features depend on browser/OS behavior (e.g., context menu, geolocation, multiple windows, alert dialogs); exact system dialogs and browser chrome are outside the application’s control and thus out of scope for strict UI requirements.

---

3. Functional Requirements

3.1 Notation and Structure

Each functional requirement is specified with:

- Requirement ID: FR-<area>-<sequence>
- Feature Name
- Description
- Preconditions
- User Actions
- Expected System Behavior
- Validation and Error Handling

For brevity, requirements are grouped by feature/module. Within each module, requirements are atomic and testable (except where explicitly describing inherently probabilistic outcomes like randomness).

3.2 Global / Cross-Page Requirements

FR-G-01 – Home Page – Examples Listing

- Feature Name: Home Page Examples List
- Description: Display a list of links to all available example pages.
- Preconditions: User accesses https://the-internet.herokuapp.com/.
- User Actions:
    - Load the root URL.
- Expected System Behavior:
    - The page shall display a heading “Welcome to the-internet” and a subheading “Available Examples”.
    - The page shall present a vertical list of example names, each rendered as a hyperlink.
    - Each hyperlink text shall match the example name (e.g., “Checkboxes”, “File Upload”, “Typos”).
- Validation / Error Handling:
    - There is no explicit validation; the page must render successfully without requiring input.

FR-G-02 – Global Footer

- Feature Name: Footer Attribution
- Description: Provide a consistent footer section attributing the site to Elemental Selenium on most example pages.
- Preconditions: User navigates to an example page (e.g., /checkboxes, /upload, /dropdown, /inputs, /download).
- User Actions:
    - Scroll to the bottom of the example page.
- Expected System Behavior:
    - The page shall display a footer containing the text “Powered by Elemental Selenium”.
- Validation / Error Handling:
    - No user input; footer rendering must not interfere with the primary feature.

3.3 Selected Feature Modules

Below, detailed requirements are specified for representative and higher-interaction modules. Remaining modules are covered in section 3.4 at a concise but still precise level.

3.3.1 Checkboxes (/checkboxes)

Source text: “Checkboxes checkbox 1 checkbox 2 Powered by Elemental Selenium”.

FR-CB-01 – Rendering of Checkboxes

- Description: Display two checkboxes with labels.
- Preconditions: User navigates to /checkboxes.
- User Actions:
    - Load /checkboxes.
- Expected Behavior:
    - The page shall display at least two checkbox input controls, labeled “checkbox 1” and “checkbox 2” or equivalent nearby text.
- Validation / Error Handling:
    - The page shall load without requiring user input.

FR-CB-02 – Checkbox State Toggle

- Description: Allow the user to toggle checkbox states.
- Preconditions: /checkboxes is loaded and at least one checkbox is visible.
- User Actions:
    - Click a checkbox.
- Expected Behavior:
    - On click, the target checkbox shall toggle its checked/unchecked state according to browser default behavior.
- Validation / Error Handling:

    - No explicit error messages. Browser default behavior applies.

3.3.2 Form Authentication / Login (/login)

Source text: “This is where you can log into the secure area. Enter tomsmith for the username and SuperSecretPassword! for the password. If the information is wrong you should see error messages. Username, Password, Login.”

FR-FA-01 – Login Form Rendering

- Description: Display username and password fields with a Login button.
- Preconditions: User navigates to /login.
- User Actions:
    - Load /login.
- Expected Behavior:
    - The page shall display a heading indicating it is a “Login Page”.
    - The page shall display an input field labeled “Username”.
    - The page shall display an input field labeled “Password”.
    - The page shall display a “Login” button.
- Validation / Error Handling:
    - No validation on initial page load.

FR-FA-02 – Successful Login with Valid Credentials

- Description: Grant access to a secure area when valid credentials are provided.
- Preconditions:
    - /login is loaded.
- User Actions:
    - Enter “tomsmith” in the Username field.
    - Enter “SuperSecretPassword!” in the Password field.
    - Click the “Login” button.
- Expected Behavior:
    - The system shall authenticate the user as successful when the username is “tomsmith” and password is “SuperSecretPassword!”.
    - Upon success, the system shall navigate to a “secure area” page (e.g., via URL change /secure or equivalent) that indicates successful login. (The exact text on that page is inferred only as “secure area” from the source and is not otherwise constrained.)
- Validation / Error Handling:
    - No error message shall be displayed for the above valid credentials.

FR-FA-03 – Error Handling for Invalid Credentials

- Description: Display error messages when login information is incorrect.
- Preconditions: /login is loaded.
- User Actions:
    - Enter any username and/or password combination other than the known valid tomsmith / SuperSecretPassword!.

    - Click the “Login” button.
- Expected Behavior:
    - The system shall display an error message indicating that the information is wrong.
    - The system shall not navigate to the secure area page.
- Validation / Error Handling:
    - The error message shall be visually associated with the login form (exact text is not constrained beyond indicating wrong information).

3.3.3 Dropdown List (/dropdown)

Source text: “Dropdown List – Please select an option – Option 1 – Option 2.”

FR-DD-01 – Dropdown Rendering

- Description: Display a selectable dropdown list with two options.
- Preconditions: User navigates to /dropdown.
- User Actions:
    - Load /dropdown.
- Expected Behavior:
    - The page shall display a descriptive heading “Dropdown List”.
    - The page shall present a dropdown (select element) with a default prompt “Please select an option” or equivalent text.
    - The dropdown shall include at least “Option 1” and “Option 2” as selectable options.
- Validation / Error Handling:
    - No validation is applied until user selection; no error states are described.

FR-DD-02 – Option Selection

- Description: Allow the user to change the dropdown selection.
- Preconditions: /dropdown is loaded, dropdown visible.
- User Actions:
    - Click the dropdown and select an option (Option 1 or Option 2).
- Expected Behavior:
    - The dropdown’s visible selection shall update to show the chosen option.
- Validation / Error Handling:
    - No error message or further navigation is required; selection is accepted silently.

3.3.4 Dynamic Controls (/dynamic_controls)

Source text: “This example demonstrates when elements (e.g., checkbox, input field, etc.) are changed asynchronously.”

FR-DC-01 – Display of Asynchronous Controls

- Description: Provide an example page demonstrating asynchronous changes to elements such as checkboxes and input fields.

- Preconditions: User navigates to /dynamic_controls.
- User Actions:
    - Load /dynamic_controls.
- Expected Behavior:
    - The page shall display descriptive text explaining that elements are changed asynchronously.
    - The page shall display at least one control (such as a checkbox or input field) that may be dynamically enabled, disabled, added, or removed after user actions.
- Validation / Error Handling:
    - Exact state transitions and timing are not constrained beyond the statement that changes are asynchronous.

3.3.5 Dynamic Loading (/dynamic_loading)

Source text: Describes two examples: “Example 1: Element on page that is hidden. Example 2: Element rendered after the fact.”

FR-DL-01 – Dynamic Loading Examples Overview

- Description: Present two separate examples of dynamically loaded page elements.
- Preconditions: User navigates to /dynamic_loading.
- User Actions:
    - Load /dynamic_loading.
- Expected Behavior:
    - The page shall display heading “Dynamically Loaded Page Elements”.
    - The page shall provide two example links or sections:
        - Example 1: Element on page that is hidden.
        - Example 2: Element rendered after the fact.
- Validation / Error Handling:
    - No validation on overview page; interactions occur within examples (not fully visible in provided HTML).

3.3.6 File Upload (/upload)

Source text: “File Uploader – Choose a file on your system and then click upload. Or, drag and drop a file into the area below.”

FR-UP-01 – File Uploader Rendering

- Description: Provide a file selection and upload capability.
- Preconditions: User navigates to /upload.
- User Actions:
    - Load /upload.
- Expected Behavior:
    - The page shall display heading “File Uploader”.

    - The page shall display a file input control allowing the user to choose a file from their system.
    - The page shall display an “Upload” button to initiate upload.
    - The page may present a drag-and-drop area for file selection per the descriptive text.
- Validation / Error Handling:
    - If the user attempts to submit without selecting a file, browser default file input behavior applies (no additional constraints observable).

FR-UP-02 – Upload Action

- Description: Allow the user to upload a file and display confirmation.
- Preconditions:
    - /upload is loaded.
    - The user has selected a file using the file input or drag-and-drop.
- User Actions:
    - Click the “Upload” button.
- Expected Behavior:
    - The system shall accept the selected file and navigate to or update the page to reflect that the file has been uploaded (confirmation text behavior is inferred from typical uploader patterns but not explicitly described in the snippet; only that user is instructed to click upload).
- Validation / Error Handling:
    - No explicit error text is provided in the HTML; any file-related errors are outside observable scope.

3.3.7 JavaScript Alerts (/javascript_alerts)

Source text: “JavaScript Alerts – Here are some examples of different JavaScript alerts... Click for JS Alert / Click for JS Confirm / Click for JS Prompt. Result:”.

FR-JA-01 – Alerts Page Rendering

- Description: Present buttons that trigger different types of JavaScript dialogs.
- Preconditions: User navigates to /javascript_alerts.
- User Actions:
    - Load /javascript_alerts.
- Expected Behavior:
    - The page shall display heading “JavaScript Alerts”.
    - The page shall display three clickable controls, labeled:
        - “Click for JS Alert”
        - “Click for JS Confirm”
        - “Click for JS Prompt”
    - The page shall display a “Result:” label where outcome text may appear.
- Validation / Error Handling:
    - No validation until a button is clicked.

FR-JA-02 – Basic Alert Trigger

- Description: Trigger a simple JavaScript alert.
- Preconditions: /javascript_alerts is loaded.
- User Actions:
    - Click the “Click for JS Alert” button.
- Expected Behavior:
    - The browser shall display a JavaScript alert dialog. (Exact text in the dialog is not visible in the HTML; only that the button is intended to show an alert.)
- Validation / Error Handling:
    - Once dismissed by the user, any status may be reflected in the “Result:” area; specific text is not constrained.

FR-JA-03 – Confirm and Prompt Triggers

- Description: Trigger a confirm and a prompt dialog respectively.
- Preconditions: /javascript_alerts is loaded.
- User Actions:
    - Click “Click for JS Confirm”.
    - Click “Click for JS Prompt”.
- Expected Behavior:
    - Clicking “Click for JS Confirm” shall display a JavaScript confirm dialog.
    - Clicking “Click for JS Prompt” shall display a JavaScript prompt dialog.
    - The “Result:” area may update to reflect user actions; specific text not constrained by source.
- Validation / Error Handling:
    - Dialog handling (OK/Cancel/value) is governed by browser; the page should not throw additional errors.

3.3.8 Drag and Drop (/drag_and_drop)

Source text is minimal: “Drag and Drop Powered by Elemental Selenium”.

FR-DDP-01 – Drag and Drop Example Rendering

- Description: Display a drag-and-drop area with draggable targets.
- Preconditions: User navigates to /drag_and_drop.
- User Actions:
    - Load /drag_and_drop.
- Expected Behavior:
    - The page shall include a region that visually suggests drag-and-drop functionality.
- Validation / Error Handling:
    - Behavior beyond rendering (e.g., success state, swapped positions) is not specified in the snippet and is not constrained here.

3.3.9 Sortable Data Tables (/tables)

Source text: Data tables with Example 1 and Example 2, containing Last Name, First Name, Due, Web Site, Action columns with multiple rows and "edit delete" actions.

FR-TB-01 – Data Tables Rendering

- Description: Display at least two table examples with user data.
- Preconditions: User navigates to /tables.
- User Actions:
    - Load /tables.
- Expected Behavior:
    - The page shall display heading "Data Tables".
    - The page shall describe "Example 1" and "Example 2".
    - Each example shall render a table with the following column headers:
        - Last Name
        - First Name
        - Due
        - Web Site
        - Action
    - Each table shall include at least the rows shown in the HTML snippet (Smith, Bach, Doe, Conway, etc.) with associated values.
- Validation / Error Handling:
    - No validation; this is a static data display by default.

FR-TB-02 – Row Actions (Edit/Delete) Representation

- Description: Provide "edit" and "delete" action controls in each row.
- Preconditions: /tables is loaded.
- User Actions:
    - View any table row in Example 1 or Example 2.
- Expected Behavior:
    - Each row shall include an "edit" and a "delete" action in the Action column.
- Validation / Error Handling:
    - The snippet does not show resulting behavior after clicking these links; therefore, no behavior beyond rendering is constrained.

3.3.10 Notification Messages (/notification_message_rendered)

Source text: "Notification Message... The message displayed above the heading is a notification message... Click here to load a new message."

FR-NM-01 – Notification Message Rendering

- Description: Display a notification message above the heading.
- Preconditions: User navigates to /notification_message_rendered.
- User Actions:
    - Load /notification_message_rendered.

- Expected Behavior:
    - The page shall display heading “Notification Message”.
    - The page shall display a notification message above the heading, describing an outcome such as “Action successful” or “Action unsuccessful, please try again” (examples given).
- Validation / Error Handling:
    - The message content may vary; only that a message is shown is required.

FR-NM-02 – Load New Notification Message

- Description: Allow the user to load a new notification message.
- Preconditions: /notification_message_rendered is loaded.
- User Actions:
    - Click the link labeled “Click here to load a new message.”
- Expected Behavior:
    - On each click, the system shall display a new notification message (text potentially different from previous message).
- Validation / Error Handling:
    - No explicit error behavior is described if a new message cannot be loaded; the site should not crash or produce visible runtime errors.

3.3.11 Entry Ad (/entry_ad)

Source text: “Displays an ad on page load. If closed, it will not appear on subsequent page loads. To re-enable it, click here. Close.”

FR-EA-01 – Entry Ad Display on First Load

- Description: Automatically display an advertisement modal when the page loads.
- Preconditions: User navigates to /entry_ad for the first time or when the ad is enabled.
- User Actions:
    - Load /entry_ad.
- Expected Behavior:
    - On page load, the system shall display an advertisement (modal or overlay) on the page.
- Validation / Error Handling:
    - The ad shall be clearly distinguishable from the background content.

FR-EA-02 – Close Ad and Prevent Reappearance

- Description: Allow the user to close the ad and suppress it on subsequent page loads.
- Preconditions: Ad is visible on /entry_ad.
- User Actions:
    - Click the “Close” control associated with the ad.
    - Reload or revisit /entry_ad.

- Expected Behavior:
    - The ad shall be dismissed from view once “Close” is clicked.
    - On subsequent page loads, the ad shall not appear (as long as the setting remains active).
- Validation / Error Handling:
    - No error messages are required; ad simply does not show again.

FR-EA-03 – Re-enable Ad

- Description: Allow the user to re-enable the ad.
- Preconditions:
    - Ad has been previously closed and is not currently shown.
- User Actions:
    - Click the link or control labeled “click here” associated with “To re-enable it, click here.”
- Expected Behavior:
    - After re-enabling, subsequent visits to /entry_ad shall again display the ad on page load.
- Validation / Error Handling:
    - No explicit error handling is described.

3.3.12 Typos (/typos)

Source text: “This example demonstrates a typo being introduced. It does it randomly on each page load. Sometimes you'll see a typo, other times you won,t.”

FR-TY-01 – Random Typo Behavior

- Description: Randomly include or exclude a typo in the displayed text on page load.
- Preconditions: User navigates to /typos.
- User Actions:
    - Load /typos.
    - Optionally refresh the page multiple times.
- Expected Behavior:
    - The page shall display descriptive text explaining that a typo may appear randomly.
    - On some page loads, at least one word or punctuation shall be incorrect (e.g., “won,t”).
    - On other loads, the text may render without such typo.
- Validation / Error Handling:
    - There is no validation or error message; randomness is the core behavior.

3.3.13 Add/Remove Elements (/add_remove_elements/)

Source text: “Add/Remove Elements Add Element Powered by Elemental Selenium”.

FR-ARE-01 – Add Element Button

- Description: Render a button for dynamically adding elements.
- Preconditions: User navigates to /add_remove_elements/.
- User Actions:
    - Load /add_remove_elements/.
- Expected Behavior:
    - The page shall display heading “Add/Remove Elements”.
    - The page shall display a button labeled “Add Element”.
- Validation / Error Handling:
    - No validation at initial load.

FR-ARE-02 – Add Element Action

- Description: Add a new element to the page when the “Add Element” button is clicked.
- Preconditions: /add_remove_elements/ is loaded.
- User Actions:
    - Click “Add Element”.
- Expected Behavior:
    - On each click, the system shall add a new button or UI element (commonly “Delete” in this example, although label not shown in snippet; exact label is not constrained).
- Validation / Error Handling:
    - No errors are expected; the system shall allow multiple additions.

FR-ARE-03 – Remove Added Elements

- Description: Allow removal of dynamically added elements.
- Preconditions: At least one dynamically added element is present.
- User Actions:
    - Click the control associated with each dynamically added element (commonly a Delete button).
- Expected Behavior:
    - The clicked element shall be removed from the DOM and no longer be visible.
- Validation / Error Handling:
    - No error message is necessary.

3.3.14 Disappearing Elements (/disappearing_elements)

Source text: “This example demonstrates when elements on a page change by disappearing/reappearing on each page load.”

FR-DE-01 – Element Disappearance/Reappearance Across Loads

- Description: On each page load, certain page elements may appear or disappear.
- Preconditions: User navigates to /disappearing_elements.
- User Actions:

    - Load the page multiple times (including refresh).
- Expected Behavior:
    - The page shall display descriptive text explaining elements may disappear/reappear on each load.
    - On some loads, certain menu items or elements shall be visible; on other loads, at least one of them shall not be present.
- Validation / Error Handling:
    - No error message is required; absence/presence is by design.

3.3.15 Hovers (/hovers)
Source text: "Hovers – Hover over the image for additional information – name: user1 / View profile, etc."
FR-HV-01 – Hover Targets Rendering

- Description: Display multiple user images with hidden details revealed on hover.
- Preconditions: User navigates to /hovers.
- User Actions:
    - Load /hovers.
- Expected Behavior:
    - The page shall display a heading "Hovers".
    - The page shall display at least three user images.
    - Each user image shall be associated with a set of hidden details:
        - name: user1 (or user2, user3)
        - A link labeled "View profile"
- Validation / Error Handling:
    - On initial load, details may be hidden until hover; no error messages required.

FR-HV-02 – Hover-to-Reveal Behavior

- Description: Reveal user details when hovering over each image.
- Preconditions: /hovers loaded, images visible.
- User Actions:
    - Move the mouse pointer over an image.
- Expected Behavior:
    - While hovering over an image, the corresponding user's name and "View profile" link shall be visible.
    - When the mouse leaves the image area, the details may be hidden again.
- Validation / Error Handling:
    - No error messages; purely UI behavior.

3.4 Other Example Modules (Concise Functional Requirements)
For completeness, the following modules are described with concise but testable requirements, based on their descriptive text.

3.4.1 A/B Test (/abtest)

- FR-AB-01: The page shall describe A/B testing or split testing as a way to test different versions of pages to determine which works best.
- FR-AB-02: The page content may vary to reflect different versions; the heading "No A/B Test" is present in the inspected variant.

3.4.2 Dynamic Content (/dynamic_content)

- FR-DCNT-01: The page shall load new text and images on each page refresh, demonstrating changing content.
- FR-DCNT-02: When ?with_content=static is appended to the URL or the "click here" link is used, some of the content shall become static.

3.4.3 Status Codes (/status_codes)

- FR-SC-01: The page shall explain HTTP status codes and list standard codes (Success, Redirection, Client Error, Server Error) with a link to a complete list.

3.4.4 Inputs (/inputs)

- FR-IN-01: The page shall render a numeric input field labeled "Number".
- FR-IN-02: The field shall accept numeric entry according to browser default behavior (e.g., up/down arrows).

3.4.5 Horizontal Slider (/horizontal_slider)

- FR-HS-01: The page shall show a horizontal slider control and an associated value indicator to its right.
- FR-HS-02: When the slider is focused and moved (keyboard arrows or mouse drag), the value displayed to the right shall update accordingly.

3.4.6 Context Menu (/context_menu)

- FR-CM-01: The page shall display a box area where right-click triggers a custom context menu item called "the-internet".
- FR-CM-02: Selecting the custom context menu item shall trigger a JavaScript alert.

3.4.7 Challenging DOM (/challenging_dom)

- FR-CD-01: The page shall display a table with headers "Lorem, Ipsum, Dolor, Sit, Amet, Diceret, Action" and multiple rows with numbered values (Iuvaret0, Iuvaret1, etc.) and "edit delete" actions.
- FR-CD-02: The page shall describe that locators are intentionally difficult (unique IDs, no helpful locators, canvas element).

3.4.8 Exit Intent (/exit_intent)

- FR-EI-01: When the mouse moves out of the viewport pane, the system shall display a modal window encouraging user action (e.g., sign up).
- FR-EI-02: The modal shall include a "Close" control that dismisses it.

3.4.9 JQuery UI Menu (/jqueryui/menu)

- FR-JQM-01: The page shall display a JQuery UI menu demonstrating nested menu items controlled by hover/mouse actions.

- FR-JQM-02: The descriptive text shall explain that visibility of elements is controlled by JQuery and may not be obvious from HTML alone.

3.4.10 JavaScript Error (/javascript_error)

- FR-JE-01: The page shall contain a JavaScript error triggered on the onload event.
- FR-JE-02: The page shall describe that this can be a problem for normal JavaScript injection techniques.

3.4.11 Large & Deep DOM (/large)

- FR-LD-01: The page shall present a deeply nested DOM structure with headings “No Siblings” and “Siblings” and a large tabular layout with hundreds of cells marked by level and index (e.g., 1.1, 1.2, ... 50.50).
- FR-LD-02: The page shall describe that this layout is used to demonstrate rendering and test performance issues when DOMs are large.

3.4.12 Infinite Scroll (/infinite_scroll)

- FR-IS-01: The page shall implement infinite scrolling such that scrolling down loads additional content blocks.
- FR-IS-02: The page shall contain a link or text “next page”.

3.4.13 File Download (/download)

- FR-DL-01: The page shall present a list of files as links under the heading “File Downloader”.
- FR-DL-02: Clicking on any file name link shall initiate a download of the corresponding file via the browser.

3.4.14 Forgot Password (/forgot_password)

- FR-FP-01: The page shall display a form with an “E-mail” field and a “Retrieve password” button under heading “Forgot Password”.
- FR-FP-02: Submitting the form shall attempt to retrieve a password for the given email; further behavior is not described in the snippet and thus is not constrained.

3.4.15 Geolocation (/geolocation)

- FR-GL-01: The page shall display heading “Geolocation” and a button labeled “Where am I?”.
- FR-GL-02: Clicking “Where am I?” shall attempt to access the user’s current latitude and longitude and display them on the page.

3.4.16 Floating Menu (/floating_menu)

- FR-FM-01: The page shall show a floating menu that remains visible while the user scrolls through long text content.
- FR-FM-02: The main content shall consist of multiple long paragraphs of placeholder Latin text.

3.4.17 Shadow DOM (/shadowdom)

- FR-SD-01: The page shall demonstrate shadow DOM usage, including the text “My default text” and repeated lines like “Let’s have some different text! In a list!”.

- FR-SD-02: Some content shall be enclosed in shadow DOM such that direct DOM queries may not see it in the regular document tree.

3.4.18 Frames (/frames and related)

- FR-FR-01: /frames shall display a list of frame-related examples, including “Nested Frames” and “iFrame”.
- FR-FR-02: Selecting “Nested Frames” shall navigate to /nested_frames where multiple frames (top/bottom, left/right) are embedded.
- FR-FR-03: Selecting “iFrame” shall navigate to a page with an editable rich text area (per WYSIWYG Editor example on the home list).

3.4.19 Windows (/windows and /multiple_windows)

- FR-WIN-01: /windows shall display text “Opening a new window” and a link “Click Here”. Clicking the link shall open a new browser window or tab.
- FR-WIN-02: multiple_windows (if distinct) shall provide a similar multi-window demonstration.

3.4.20 Shifting Content (/shifting_content)

- FR-SHC-01: The page shall list three examples: “Menu Element”, “An image”, “List”, describing that elements shift a few pixels on each page load.
- FR-SHC-02: On reload, elements’ positions shall change slightly, reflecting subtle rendering shifts.

3.4.21 Infinite & Slow Resources, Secure File Download, etc.

- For examples listed on the home page but not visible in the retrieved snippets (e.g., “Slow Resources”, “Secure File Download”), the scope is limited to:
    - Rendering their headings and descriptive text similar in style to other examples.
    - Demonstrating the stated concept (e.g., slow-loading resources or secure file downloads) without specifying further behavior beyond visible descriptive content.

---

4. Non-Functional Requirements

4.1 Usability

- NFR-U-01: Page structures shall be simple and minimalistic so that intended test interactions (clicks, key presses, hovers, etc.) are easily discoverable without prior instruction.
- NFR-U-02: Each example page shall include explanatory text at or near the top that clearly states the behavior being demonstrated (e.g., “This example demonstrates a typo being introduced...”, “This example demonstrates the ever-evolving nature of content...”).
- NFR-U-03: Important interactive controls (such as “Login”, “Add Element”, “Where am I?”, “Click for JS Alert”) shall be visually distinguishable as buttons or links and labeled with descriptive text reflecting their actions.

- NFR-U-04: The visual appearance of the site shall remain intentionally simple and unobtrusive, allowing users to focus on testing behavior rather than on elaborate design details.
- NFR-U-05: The site shall provide a consistent and intuitive feel across examples; users should be able to understand new examples quickly based on prior experience within the site.

4.2 Performance

- NFR-P-01: Under normal network conditions, each example page shall render basic static content within a reasonable time frame suitable for interactive testing.
- NFR-P-02: Features aiming to demonstrate "Slow Resources" may intentionally delay loading of certain elements, but even in such cases the main page shell shall load in a way that allows testers to perceive and experiment with the delay.
- NFR-P-03: Dynamic behaviors that rely on page refresh (e.g., Dynamic Content, Typos, Disappearing Elements) shall complete new content rendering immediately after the browser completes the reload event.

4.3 Reliability

- NFR-R-01: All example pages listed under "Available Examples" on the home page shall be accessible via their respective links without resulting in HTTP errors (e.g., 404), except where specific examples are purposely demonstrating error conditions (e.g., Status Codes).
- NFR-R-02: Randomized behaviors (e.g., Typos, Dynamic Content, Notification Messages) shall not cause JavaScript errors that prevent the page from rendering correctly.
- NFR-R-03: The site shall tolerate repeated navigation, refresh, and interaction cycles without requiring a restart or leading to inconsistent, unrecoverable states.

4.4 Compatibility

- NFR-C-01: The site shall operate correctly on common modern desktop browsers that support JavaScript, HTML5 forms, and the Geolocation API (for the geolocation example).
- NFR-C-02: Context-menu, alert, confirm, and prompt examples shall behave consistent with browser-native UI patterns for those dialogs, without requiring additional plugins.
- NFR-C-03: Multiple windows and tab behaviors (e.g., /windows, /multiple_windows) shall function correctly in standard desktop browsers, opening new windows or tabs as configured by browser settings.

4.5 Maintainability and Modularity

- NFR-M-01: Each example page shall be logically independent so changes to one example do not affect functionality of others.

- NFR-M-02: Example pages shall be addressable directly by URL so that they can be used independently as test fixtures or training assets.

4.6 User Experience and Visual Appeal (Intentionally Subjective)

- NFR-UX-01: The overall aesthetic of the application should feel clean and uncluttered, with ample whitespace and minimal distractions so that users can concentrate on the functional behavior under test.
- NFR-UX-02: The typography and spacing across pages should provide a sense of consistency and calmness, avoiding abrupt changes in font size or layout that might confuse or distract users during testing activities.
- NFR-UX-03: Color usage should remain subtle and neutral, conveying a professional demo environment rather than a marketing site, while still making controls and headings easy to visually scan.

---

5. Edge Cases and Negative Scenarios

This section enumerates key edge and negative scenarios inferred from the design of each example.

5.1 Random and Non-Deterministic Behaviors

- Typos (/typos):
  - Edge: Users may refresh the page multiple times without seeing a typo due to randomness. Test cases should accommodate that a typo is not guaranteed on every load.
  - Negative: Automation relying on exact text match may intermittently fail; such fragility is the point of this example.
- Dynamic Content (/dynamic_content):
  - Edge: Text and images may change on every refresh; tests must not assert exact content unless ?with_content=static is present.
- Notification Messages (/notification_message_rendered):
  - Edge: Different messages can appear after each “Click here to load a new message”. A test asserting any one specific string may fail intermittently.
- Disappearing Elements (/disappearing_elements):
  - Edge: Certain menu items may be absent on a given load; tests relying on their presence must consider repeated refresh or conditional logic.

5.2 Input and Form Scenarios

- Login (/login):
  - Negative: Wrong username, wrong password, or both shall result in an error message and refusal of access to secure area.
  - Edge: Empty username or password fields; behavior is not explicitly described but would be treated as incorrect information and thus produce an error.
- Forgot Password (/forgot_password):

    - Edge: Email field left blank, invalid email format; no explicit validation behavior is described; application may accept and process without specific UI error.
- Inputs (/inputs):
    - Edge: Non-numeric characters entered into the numeric field; actual response will depend on browser default behavior (e.g., ignoring characters), which should not break the page.
- File Upload (/upload):
    - Negative: Submit without choosing a file; browser may prohibit submission or show default error, but application itself does not specify custom messages.
    - Edge: Uploading large files or unsupported formats; no constraints described; behavior may vary but must not crash the page.

5.3 Interaction and Control Scenarios

- JavaScript Alerts (/javascript_alerts):
    - Negative: User cancels confirm or dismisses prompt without entry; “Result:” output should still render without script errors.
    - Edge: Multiple rapid clicks on alert buttons; repeated dialogs should not break the page.
- Context Menu (/context_menu):
    - Edge: Right-click outside the specified box should present only default browser context menu, not the custom “the-internet” item.
    - Negative: If user ignores the alert triggered by context menu selection, the rest of the page should remain functional after alert dismissal.
- Hovers (/hovers):
    - Edge: Hover events on mobile/touch environments may not behave as in desktop; user may need to tap; no explicit adaptation is described, so primary test focus is desktop browsers.
- Horizontal Slider (/horizontal_slider):
    - Edge: Keyboard and mouse interactions should both be supported; repeated movement beyond extremes should clamp value at minimum or maximum.

5.4 Layout and Rendering Scenarios

- Large & Deep DOM (/large):
    - Edge: Slow rendering or heavy scroll due to large DOM; tests should ensure page remains responsive and no critical JS errors occur.
- Infinite Scroll (/infinite_scroll):
    - Negative: Scrolling beyond available dynamically loaded content should gracefully stop adding new sections without error.
- Floating Menu (/floating_menu):

  - Edge: Very fast scrolling should not detach the floating menu from its expected position or cause flickering.
- Shifting Content (/shifting_content):
  - Edge: Pixel-level shifts on each load can cause automation based on absolute positions to fail; locators should be robust to such shifts.

5.5 Browser and System Edge Cases

- Geolocation (/geolocation):
  - Negative: User denies location permission; the page should handle this gracefully (e.g., no coordinates available) without crashing.
  - Edge: Location services temporarily unavailable; a message or empty coordinates should appear rather than unhandled errors.
- Windows (/windows):
  - Edge: Pop-up blockers enabled; new window may not open; base page should still function without errors.
- JavaScript Error (/javascript_error):
  - Edge: JS error on load is intentional; tests should verify that error exists while page remains at least partially accessible for inspection.

---

6. Out-of-Scope Items

This SRS explicitly excludes the following:

1. Backend Data Storage and Business Logic
   - No requirements are specified for databases, persistence, or server-side business rules beyond what is observable in the UI.
   - User accounts, password retrieval logic, or content management processes behind the examples are out of scope.
2. Security Policies Beyond Examples
   - System-wide authentication, authorization, and security policies (e.g., OWASP controls, encryption) are not defined beyond the “Form Authentication” example context.
   - Secure file downloads’ server configuration and access control are not specified.
3. Mobile-Specific Behavior
   - Responsive design, mobile breakpoints, and touch-specific optimizations are not covered. All requirements assume desktop-class browsers.
4. Exact Visual Design Specifications
   - Pixel-perfect positioning, colors, fonts, font sizes, and spacing beyond high-level usability and consistency expectations are not defined.
   - Branding, logos, and graphic design guidelines are out of scope.
5. Accessibility Compliance

    - Detailed accessibility requirements (ARIA roles, screen reader support, keyboard-only navigation beyond the demonstrated examples) are not defined.
6. Internationalization and Localization
    - The site content is observed only in English. No requirements are defined for translations, date/number formats, or multi-language support.
7. Undocumented Interactive Behavior
    - For any example where click-output or post-action result is not observable from the HTML snippets (e.g., exact result text after uploading, table sorting interactions), such behavior is not specified in this SRS to avoid unsupported assumptions.

This SRS captures the observable structure, behaviors, and key edge cases of the "The Internet" example site as deployed at https://the-internet.herokuapp.com/, using formal, atomic, and testable requirements wherever supported by the live UI and embedded text.