Planning Analysis Sheet: RENTOZY Property Listings Platform

Website Link - https://tanvi-bagwe.github.io/rentozy/

Repository Link - https://github.com/Tanvi-Bagwe/rentozy

This document outlines the design, development process, and technical implementation of the RENTOZY project. RENTOZY is a static, single-page application built to showcase a functional property listings platform using fundamental web technologies.

1.**Design Decisions and User Experience**
From the beginning, my goal for RENTOZY was to create a simple, clean, and intuitive user experience. The design choices were made with the end-user in mind, focusing on clarity and ease of use.

- **Single-Page Layout:** I decided to build RENTOZY as a single-page application (SPA). This was a deliberate choice to prevent full page reloads and create a smoother browser experience. When a user navigates between "Home," "Listings," and "Gallery," the page content changes instantly, which feels fast and modern.

- **Visual Hierarchy:** The layout is structured to draw the user's eye to the most important elements. The "Property Listings" section, for example, uses a card-based layout to make each property easy to read and digest.

- **Forms and Interactivity:** I wanted the "Add Property" feature to be as straightforward as possible. The form is initially hidden to reduce clutter, and clear error messages are displayed directly beneath the form fields if validation fails. The search and clear buttons are visually distinct, using color to communicate their primary and secondary functions.

- **Responsive Design:** I included the viewport meta tag and designed the layout with a flexible CSS approach. This ensures the site adapts gracefully to different screen sizes, from desktop monitors to mobile phones, making it accessible to a wide audience.

2. **Development Process and Implementation**
The project was built using a modular approach, separating the code into logical files to improve organization and maintainability.

- **HTML Structure:** The HTML serves as the foundation of the application. I focused on using semantic tags like <header>, <nav>, <section>, and <footer> to give the document a logical structure. This is not only good for SEO but is also crucial for accessibility, allowing screen readers to better interpret the page content. I also made sure every form input was properly linked to its corresponding <label> using for and id attributes.

- **CSS Styling:** My CSS was broken down into separate files for each major component of the site. This modular approach made it easier to manage styles and prevent conflicts between different sections. For example, all styles related to the listing cards are contained within listings.css, while all navigation-specific styles are in header.css.

- **JavaScript Interactivity:** JavaScript is the engine that brings RENTOZY to life. I separated the functionality into three main scripts:
  main.js (Page Navigation): This script handles the SPA functionality. It uses a hashchange event listener to detect when the URL's hash changes (e.g., from /#home to /#listings). The

loadPage() function then gets the correct id from the hash and shows only the corresponding <section>, hiding all others. This simple but effective technique is what gives the site its dynamic feel.

- **listings.js (Core Functionality):** This is the heart of the application.
  Data Management: The listingsData array acts as a simple client-side "database" for all the property information.
  Rendering: The displayHouses() function is responsible for dynamically generating and rendering the HTML for all property cards based on the data array it receives. This function is reused for both the initial page load and for displaying search results.
  Form Logic: The submitForm() function handles the new property form. It contains a robust set of validation checks for each field, ensuring the data is correct before it's added to the listingsData array.
  Search and Filter: The searchListings() function is a key feature. It takes the user's input, converts it to lowercase, and then uses the .filter() method to create a new array containing only the properties that match the search term in their title, location, or description.

- **gallery.js (Image Slider):** This script implements a simple image slider. It uses an array of image paths and a current index variable. When the "next" or "previous" buttons are clicked, the index is incremented or decremented, and the src attribute of the main gallery image is updated. I made sure to include logic to wrap the index around to the start or end of the array, so the user can continuously cycle through the images.

3. **Challenges Faced and Solutions**

As with any project, I encountered a few challenges along the way that helped me learn and grow as a developer.

- **SPA Navigation (Hide and Show):** One of the first challenges was figuring out how to build a Single-Page Application without a complex library. My initial thought was to use multiple functions to manually show and hide each section, like showListings() and showGallery(). I quickly realized this would become a messy and unscalable approach. The solution was to use a single, unified approach. I implemented a loadPage() function that uses window.location.hash to determine the current page. It would get the ID of the new section, hide the previously active one, and then display the new one. This approach made the navigation logic much cleaner and more scalable.

- **Form Validation:** another challenge was making the form validation both comprehensive and user-friendly. Initially, I was only checking if the fields were empty. I quickly realized I also needed to validate the length of the string inputs. A bigger challenge was preventing duplicate property titles; I solved this by iterating through the listingsData array to check for an existing title before adding a new property.

- **JavaScript Organization:** I started with all my JavaScript code in one file, which quickly became messy and hard to read. I learned that separating the code into main.js, listings.js, and gallery.js was a much cleaner and more professional approach. This made debugging and feature development far easier.
- **Avoiding Redundant Code:** When I implemented the search feature, my first thought was to write a whole new function to display the filtered results. I then realized I could simply pass the filtered array to my existing displayHouses() function, which made my code more efficient and reusable. This was a great lesson in modular and reusable code.

Overall, developing RENTOZY was a fantastic experience that solidified my understanding of building a functional and interactive web application from the ground up.

Page Structure &Information

1. Rentozy – Home - index.html (Home Page) (Single Page App)

- Welcome banner with the "RENTOZY" title and mission statement.
- Navigation menu in the header to all site sections (Home, Listings, Gallery).
- Homepage description with quick access to the listings.
- Contact and social media information in the footer.

2. Property Listings - index.html

- Display of all property listings in a grid format.
- A button to open the "Add Property" form.
- A comprehensive "Add Property" form with client-side validation.
- Search functionality with Search and Clear buttons

3. Property Gallery - index.html

- Main image display for viewing property photos.
- Navigation controls (Next and Previous buttons) to cycle through images.

List of Sources

Facts & Text Content:

Flaticon (flaticon.com) - Logos and Icons

Images - https://unsplash.com/

Tools Used

Visual Studio Code - IDE for development

Draw.io - Online tool for wireframing and Flowchart

Microsoft Word – For creating planning analysis sheet.

Browsers – For testing the website performance Chrome, Edge, Safari