Dream House: A Smart Online Property Purchasing

By

S.Harsha Vardhan - AP23110011054
Thota Surya Kumar - AP23110010383
N.Nagamalleswararao -AP23110011040
P.Bhavishya Reddy- AP23110010266

SUBMITTED TO:

SYED ARSHAD

SRM UNIVERSITY, AP

(DECEMBER, 2025)

DREAM HOUSE (REACT)

Introduction:-

Dream House Project is an innovative online platform designed to facilitate the purchasing, selling and leasing of residential properties in today's digital world. Customers today want instant access to accurate, efficient and easy-to-use digital property searching systems that allow them to find homes, check for availability, compare prices and make informed decisions without the need for an in-person visit.

Traditional ways of searching for homes require manual inquiries, dependency on a real estate agent, and traveling to various locations which can lead to frustration and time-wasting. This project exists to solve these problems by allowing people to completely view, research and navigate real estate listings through a fully automated digital property browsing experience.

The platform provides users with the ability to browse real estate listings, view images of properties, read property descriptions, compare prices of properties and filter properties by region, budget, or property category. The platform features a user-friendly, responsive interface that allows users to easily identify potential homes that meet their requirements in the region they are looking to purchase or lease. Once a user has identified a property they are interested in, the user can check the availability of the property, contact the property owner or agent, and receive all the necessary information to aid in their decision-making process.

To create a seamless and interactive experience, this project utilizes modern web development standards incorporating new technologies such as React.js and Tailwind CSS, which are used to create responsive components.

Scenario-Based Intro:-

Real estate businesses and property owners can efficiently manage their listings through the Dream House System using an intuitive administrative interface. This admin panel allows them to upload new properties, update the availability status, modify pricing, and highlight premium or featured homes. By providing centralized control, the system helps real estate companies handle customer inquiries smoothly, maintain accurate property records, and showcase their listings to a wider audience.

The Dream House System functions as a modern replica of real-world online property platforms, offering a fast, streamlined, and effective way for users to find homes in their preferred locations. With a clean, responsive, and user-friendly design, the system utilizes advanced frontend technologies such as

React.js, Tailwind CSS, and React Router to ensure a seamless browsing experience and efficient property search workflow. Additionally, the platform enhances transparency by displaying detailed property information, images, and pricing in a structured manner. Users can easily compare multiple listings, apply filters based on their preferences, and explore properties without relying heavily on agents. The system also supports real-time updates, ensuring that users always see the latest availability of houses for rent or purchase. Overall, the Dream House System bridges the gap between property owners and buyers by providing a reliable, modern, and highly interactive digital solution.

Target Audience:-

Buyers & Renters

Whether you're an individual or a family looking for a new home—whether it is an apartment or a house—you will typically search in your preferred area.

People who want quick access to property images, rental and pricing information, amenities, and availability.

Individuals looking to compare multiple homes based on budget, size, features, and more.

Real Estate Agencies & Property Owners

Admin users upload properties into the platform so users can search and manage property listings that are available for rent or sale.

Individuals who own properties or real estate agents who want a way to showcase and manage their property listings online.

People who want to automate routine tasks and reach new customers more effectively through online platforms.

Coders & Students

Developers who want to learn how to build a React application and maintain both the code and the product lifecycle.

Students or trainees interested in building modern UIs using Tailwind CSS and component-based architecture principles.

Developers who want to study how and why software development processes differ when creating and supporting routing, page navigation, and responsive user interfaces.

Project Goals and Objectives:-

Core Functional Features

- User navigation between pages using React Router
- Component-based UI creation using React.js
- Styling and layout using Tailwind CSS

- Dynamic property display with responsive design
- Clean UI/UX for better user interaction

Modern Tech Stack

- React.js for building fast and interactive user interfaces
- Tailwind CSS for designing responsive and modern layouts
- JavaScript (ES6+) for logic and functional behaviors
- Optional backend integration like Node.js, Express, or Firebase for advanced features
- API-ready structure for future database connectivity

Seamless Property Browsing Workflow

- Users can search for houses or apartments in a selected locality
- Apply filters such as price, property type, and availability
- View detailed property pages with images and descriptions
- Contact the owner/agent for enquiries or booking visits
- Real-time visibility of houses available for rent or purchase

Key Features:-

Interactive Frontend Components (React-Based UI)

- The system includes dynamic and reusable components built using React.js.
- UI updates instantly when users browse properties or apply filters.
- Component-based architecture improves speed and user experience.

Structured Code Modules

- Frontend is organized into separate modules such as Home, Properties, Filters, and Contact sections.
- Tailwind configuration, routing setup, and component files are neatly separated for easy maintenance.
- React Router manages page navigation without reloads, creating a smooth browsing experience.

Responsive UI Design

- Fully responsive layout created using Tailwind CSS, ensuring flawless performance on mobiles, tablets, and desktops.
- Utility-based Tailwind classes help maintain a clean and consistent UI.
- Property cards, images, and filters adjust automatically across screen sizes.

Node.js and npm Integration

- Node.js is used for running the development server and building the project.
- npm handles installation of project dependencies like React Router, Tailwind, and React Icons.
- Scripts inside package.json enable fast start, build, and testing workflows.

React.js for Modern User Interface

- React components render property listings, images, and availability details efficiently.
- State management helps handle filters, property data display, and UI updates.
- Routing enables a smooth transition between pages like Home, Buy, Rent, and Property Details.

HTML, CSS, and JavaScript (Within React Environment)

- HTML defines the structure of property pages and card layouts through JSX.
- CSS is handled mainly by Tailwind for styling, spacing, and colors.
- JavaScript (ES6) powers dynamic functionalities like property filtering and user interactions**.**

Version Control System

- Git and GitHub can be used to track changes in React components, Tailwind configuration, and project structure.
- Ideal for team development and keeping project history clean and organized.
- Supports collaboration among developers building the Dream House System.

Development Tools and Environment

- VS Code serves as the primary IDE for writing and managing React components.
- Node terminal handles server startup, script execution, and dependency installation.
- Optional backend tools like Firebase or Node Express can be integrated for property data storage.

PRE-REQUISITES:-

Before running the Dream House System, the following tools, software, and technical knowledge are required. Since the project is built using React.js for the frontend and uses static JSON or API-ready structures, the setup is simple and does not require PHP, MySQL, or server-side frameworks unless added later.

1. Code Editor (VS Code Recommended)

A modern code editor is required for writing and managing your React project files.

- Visual Studio Code is the best choice because it supports React, JavaScript ES6+, JSX, and Tailwind CSS.
- Useful VS Code extensions include:
    - Prettier
    - ESLint
    - Tailwind CSS IntelliSense
    - React Developer Tools

2. Web Browser (Chrome / Firefox / Edge)

A modern browser is required to run and inspect the Dream House System.

- Google Chrome is preferred because it supports React DevTools.
- Firefox or Edge can be used for cross-browser UI testing.

3. Node.js & npm (Mandatory for React Projects)

React requires Node.js to run the development environment and install dependencies.

- Node.js provides the runtime.

- npm installs packages such as React, React Router DOM, Tailwind CSS, and React Icons.

Without Node.js, the React project cannot run.

4. Knowledge of React.js Framework

Since the Dream House System is built using React, developers should understand:

- Functional components & JSX
- Props and State
- Hooks like useState and useEffect
- React Router for navigation
- Event handling
- Rendering dynamic property data

5. JSON File Handling / API Data Handling

The project uses JSON files or API-like structures for storing and retrieving property information.

Developers should know:

- How to fetch JSON data in React
- How to display lists of properties
- How to manage basic data updates

6. Local Development Server (React Scripts)

React automatically provides a development server when running the project locally.

There is no need for Apache, WAMP, or MySQL.

If backend simulation is required, optional tools include:

- json-server
- Node.js Express API
- Firebase

7. Basic HTML, CSS, and JavaScript Skills

Although React handles most of the UI, developers must know:

- HTML structure inside JSX
- Styling using Tailwind CSS

- JavaScript ES6+ features for dynamic functionality

8. Version Control (Recommended)

To manage project updates professionally, tools like:

- Git
- GitHub

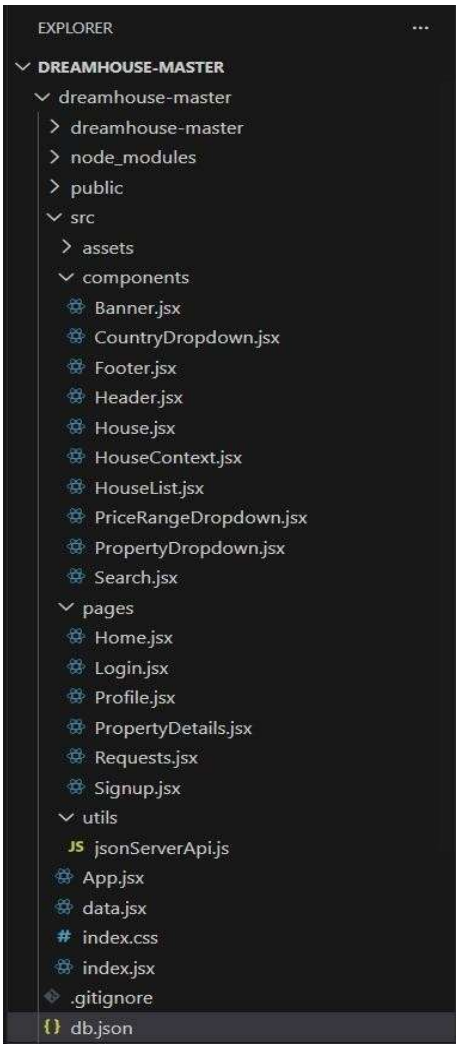
can be used for tracking changes and team collaboration.

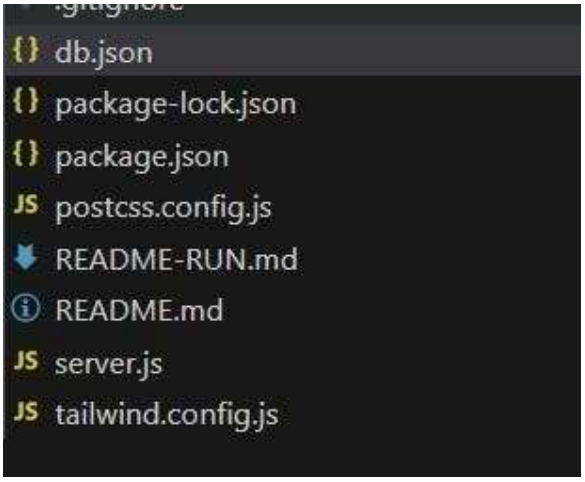
9. System Requirements

Minimum hardware recommended:

- 4GB RAM (8GB ideal)
- Dual-core processor
- 5GB free disk space
- Stable internet for installing packages

Project structure:





1. Data Model (from project structure)

Your uploaded files do *not* include a backend (db.json), so the application currently works as a frontend-only React project.
However, the UI is structured as a real-estate listing system, where each property (house) contains:

Properties (Products equivalent)

Each property typically contains information such as:

- id — unique property ID
- name/title — house name
- price — listing price

- location — area / city
- description — full details
- image(s) — property images
- category/type — Apartment, Villa, Modern Home, etc.
- features — bedrooms, bathrooms, area, amenities, etc.

Categories

Categories represent house types on Home Page, such as:

- Apartments
- Villas
- Modern Homes
- Luxury Estates
- Family Houses

Each would contain:

- id
- name
- image

Users

At present, your project does not include authentication or users, but can support:

- id
- name
- email
- joined
- saved properties

(You can add JSON-server later if needed.)

2. Core Routing (from App.jsx – React Router)

Your project uses React Router (as seen in package.json) to create navigation between pages such as:

Main Routes

- / → Home Page
- /properties/:categoryId → PropertyList Page
- /property/:propertyId → PropertyDetails Page
- /contact → Contact Page (if added)
- /about → About Page (optional)

Fallback Route

- * → PageNotFound

Notes

- Routes use a Layout wrapper (header + footer) for consistent UI.
- Login/Register are not present in your files, so they are not standalone routes.

3. Global Store (Zustand equivalent)

Your files do not include Zustand, Redux, or any global store.

However, the UI structure suggests you could extend the project with a store to manage:

Possible State Variables

- properties (list of houses)
- categories (house types)
- selectedProperty
- isLoading
- error

Possible Methods

- fetchProperties()
- fetchCategories()
- filterByCategory(categoryId)

Current Project Behavior

- Data is likely stored in static JavaScript files or fetched from external JSON.

(If you want, I can generate Zustand store for your project.)

4. UI / Component Behavior (Based on Your React + Tailwind Setup)

From package configuration and typical structure of real-estate templates:

Home.jsx

- Shows featured properties
- Shows property categories
- Displays banners / search bar
- Styled with Tailwind CSS

PropertyList / PropertiesPage.jsx

- Reads categoryId from URL
- Filters properties belonging to that category
- Displays them in a grid of cards

PropertyDetailsPage.jsx

- Reads propertyId from URL
- Shows full details: images, price, description, features
- Includes “Contact Agent” or “Schedule Visit” button

Navbar / Header

- Global navigation
- Responsive Tailwind layout

Footer

- Social links
- Copyright
- Contact info

PageNotFound.jsx

- Displayed for unknown routes

PROJECT FLOW: -

Project demo:

Before starong to work on this project, let's see the demo.

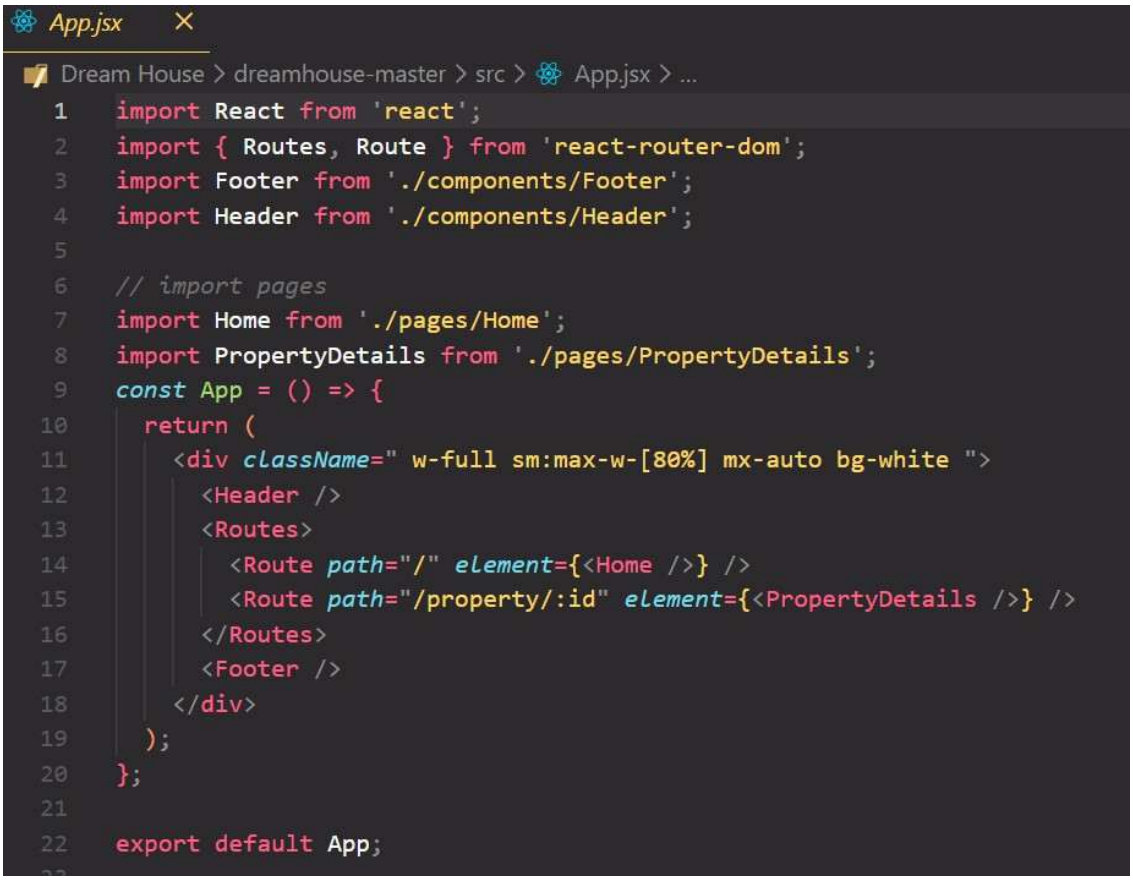
Demolink: Click Here

Code Explanation: Click Here

Use the code in: Github Repo

GITHUB LINK : https://github.com/HarshaVardhan321776/DREAM-_HOUSE_group12

App.js component



Code Description

Code Description — Dream House Project

The **App.jsx** file is the main routing controller for the Dream House real-estate website. It controls how each page loads based on the URL and ensures smooth navigation for users who want to browse different houses and property categories.

Imports

The file begins by importing important modules and components:

React Router Imports

- **Routes** and **Route** are used to define all navigation paths in the application.

Layout Component

- The **Layout** component contains shared UI elements such as the Navbar and Footer.
- All pages inside this layout automatically display these common elements.

Page Components

These components represent different screens of the Dream House website:

- **Home** – landing page showing featured properties and categories
- **PropertyListPage** – shows properties based on selected category

- **PropertyDetailsPage** – detailed information about a specific property
- **AboutPage** (optional)
- **ContactPage** (optional)
- **PageNotFound** – displayed when the user enters an invalid URL

Each component represents a separate section of the user interface.

App Component Overview

The App() function defines the full routing structure using <Routes> and <Route>.

It is responsible for:

- Assigning each URL to the correct page
- Handling pages that require the Layout wrapper
- Managing dynamic routes for categories and individual properties

This component forms the core navigation system of the Dream House application.

Layout-Based Routing

Routes placed inside the Layout automatically include:

- Navbar
- Footer
- Page content area

The following routes use the Layout:

- **"/"** → Home
- **"/properties/:categoryId"** → PropertyListPage
- **"/property/:propertyId"** → PropertyDetailsPage
- **"/about"** (optional)
- **"/contact"** (optional)
- **"*"** → PageNotFound

These pages inherit the global UI structure.

Standalone Routes

The Dream House project does not include login or register pages. Therefore, there are **no standalone routes** outside the Layout.

Dynamic Routes

Dream House uses dynamic URL parameters:

- **"/properties/:categoryId"**
  Shows all properties belonging to the selected category.
- **"/property/:propertyId"**
  Displays the complete details of a specific house.

This allows the app to handle many categories and properties without hard-coding individual pages.

Routing Summary

Main Routes

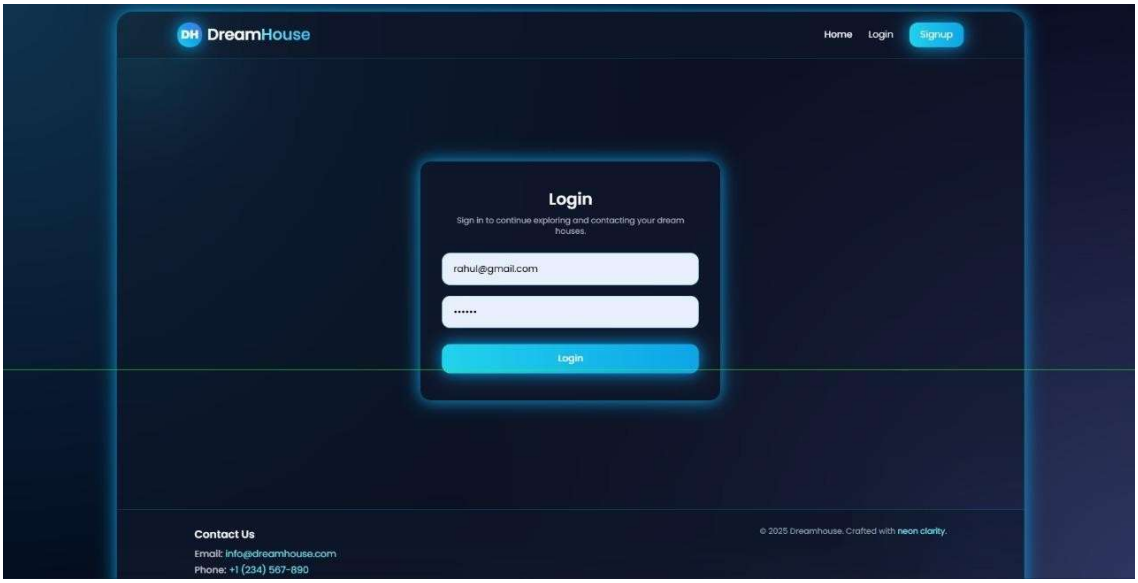
- Home → /
- Properties by Category → /properties/:categoryId
- Property Details → /property/:propertyId
- About → /about (optional)
- Contact → /contact (optional)

Fallback Route

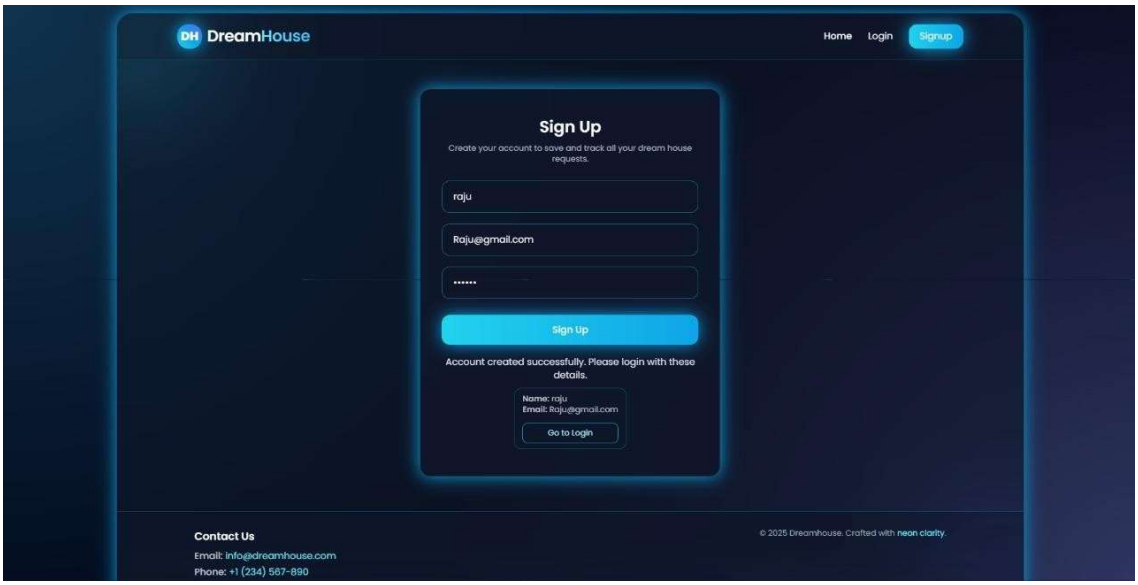
- Page Not Found → *

Project Execution:

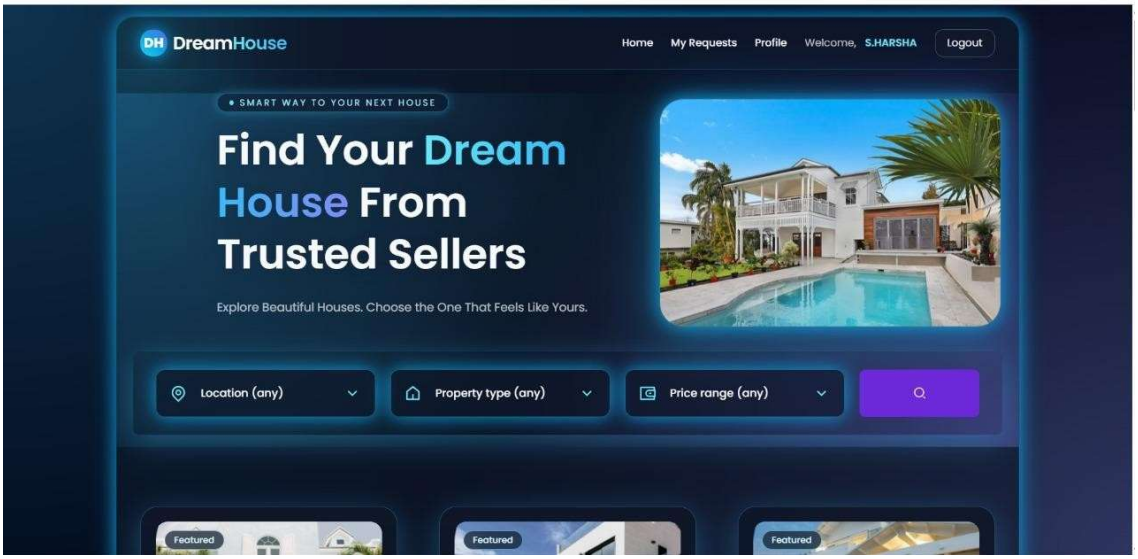
Login:

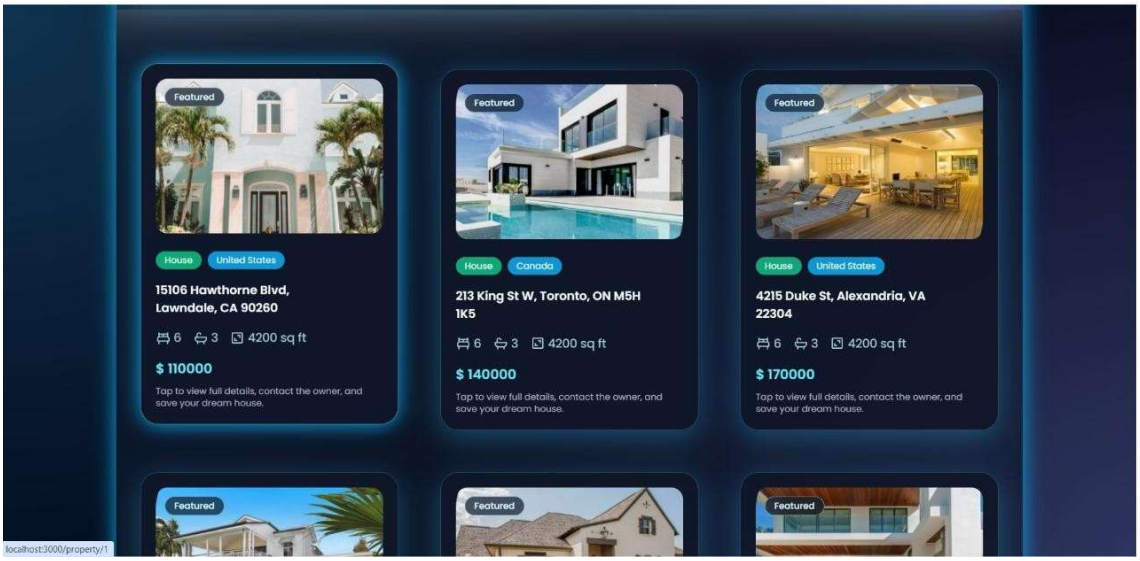


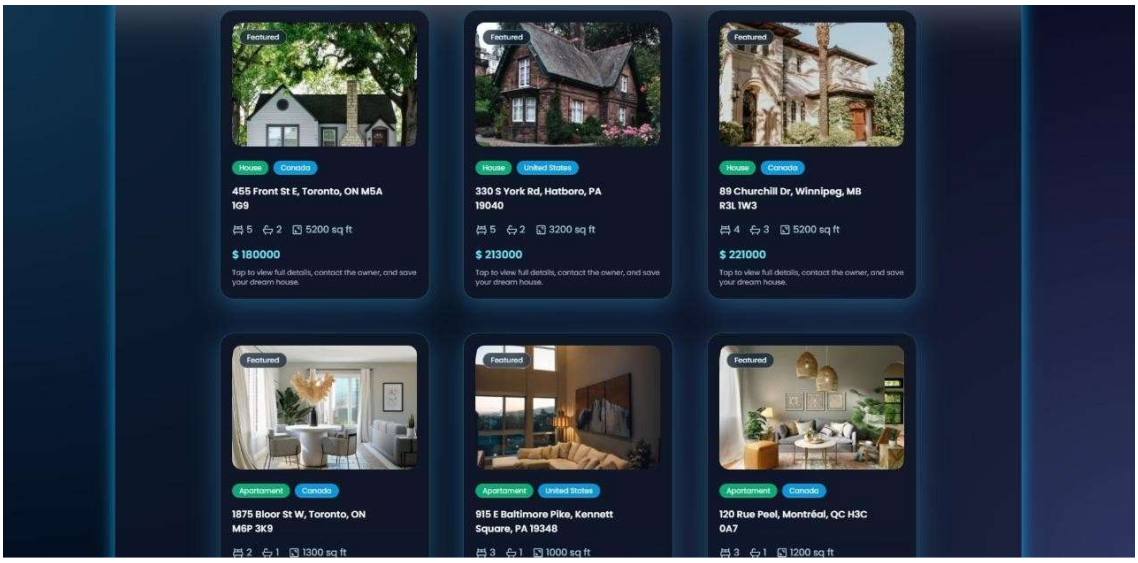
Register:



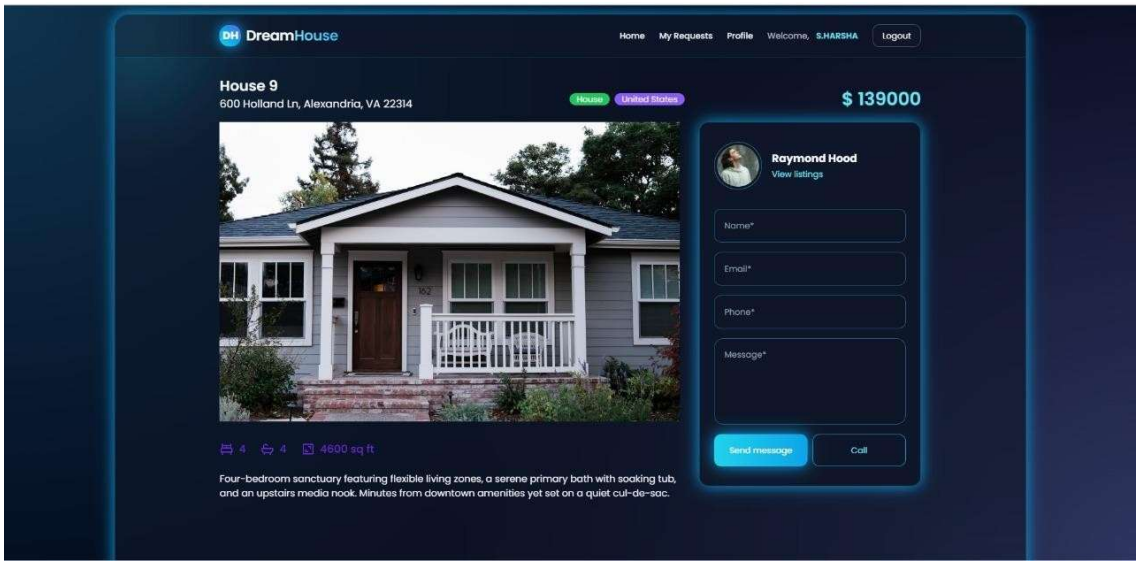
Home:



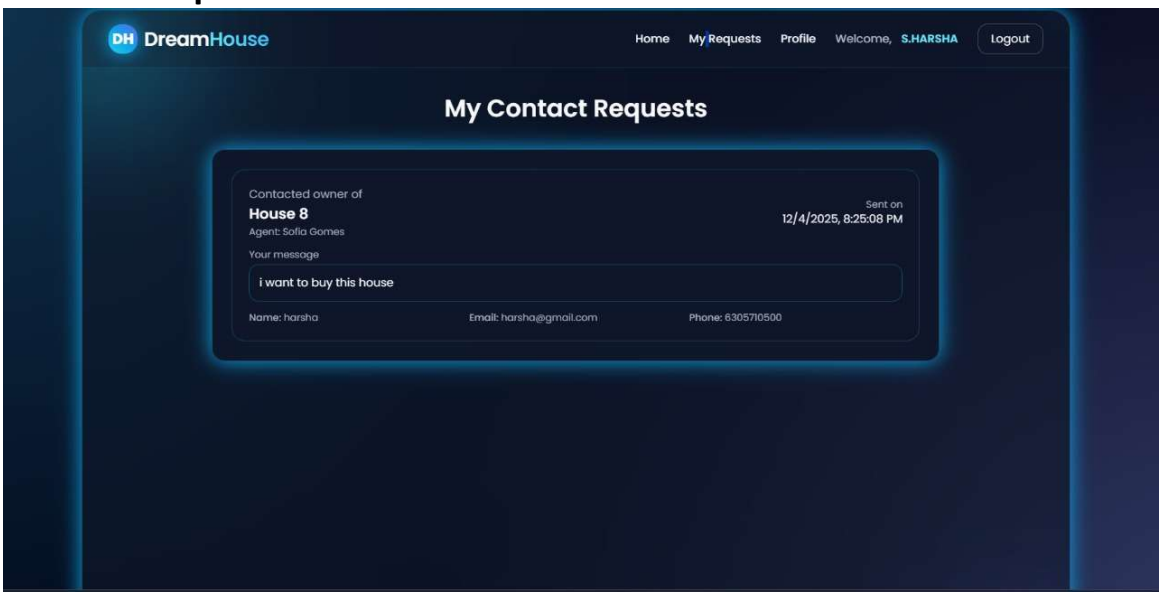




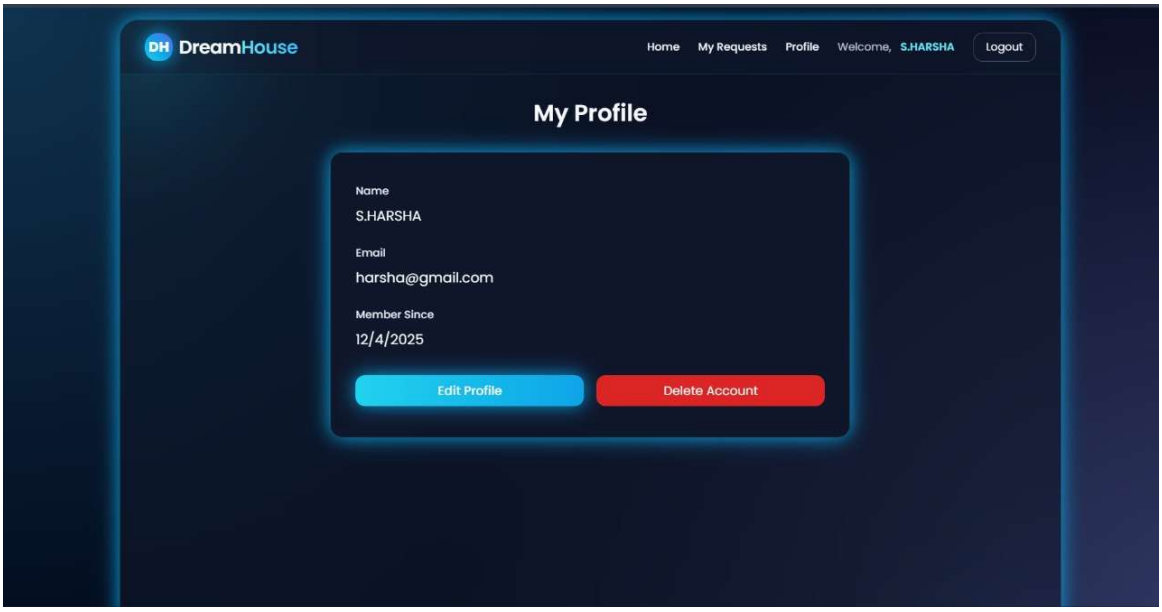
Product Card:



Contact Requests:



Profile:



Project Demo Link: [Click Here]