|  |  |  |  |
| --- | --- | --- | --- |
| **Goals:** | Crafting a Seamless Experience: Building a User-Friendly and Interactive Web Platform Tailored for UK Real Estate Enthusiasts | | |
| **Measures of Effectiveness:** | Task Completion Time:  Measure the time it takes for users to complete common tasks within the application, such as uploading documents or accessing property information.  Utilize analytics tools or custom logging to track task completion times.  Click Count:  Monitor the number of clicks required for users to accomplish specific tasks.  Aim to minimize the number of clicks needed for essential actions.  Usability Testing:  Conduct usability testing with real estate users to gather direct feedback on the application's ease of use.  Use scenarios that reflect common tasks in the real estate workflow.  User Satisfaction Surveys:  Periodically survey users to gauge their satisfaction with the application's usability.  Implement a simple and concise survey with questions related to the user interface, navigation, and overall experience. | | |
| Objectives | Workflow Planned to Meet Objectives | Packages Used | Versions & cost |
| User Authentication  ,Clear Navigation,Consistent Layout,Visual Hierarchy | We use a straightforward navigation structure with easily understandable labels  Maintain a consistent layout throughout the application for a cohesive user experience, Prioritize information based on importance, guiding users through logical flow | Backend: Node.js, Express.js, MongoDB (with AWS)  Frontend: React, Redux, Bootstrap 5,CSS3  Security: Bcrypt, JSON Web Token (JWT), OAuth,Passport.js for users authenticating using Google, facebook ,Apple. |  |
| Simple Registration: Implement a hassle-free registration process with minimal required information.  Intuitive Property  Search: Create an easy-to-use search functionality with filters for location, price range, and property type.  Interactive Maps: Include interactive maps to help users visualize property locations and nearby amenities. | (Property Registration)  Frontend (React):  react-hook-form: A performant and lightweight library for handling form validation and submission.  Backend (Node.js):  express-validator: A powerful middleware for validating and sanitizing user input, ensuring data integrity.  (search)  Frontend (React):  react-select: A versatile component library for creating searchable dropdowns and multi-select fields.  react-range-slider: A user-friendly component for implementing price range filtering.  Backend (Node.js):  mongoose-search: A full-text search extension for Mongoose, enabling efficient search queries on property data.  (MAPS):  Frontend (React):  react-map-gl: A performant and customizable React library for integrating maps with Mapbox GL JS.  Or  Google Map integration.  Backend (Node.js):  node-geocoder: A Node.js library for converting addresses into geographic coordinates, enabling map markers. |  |  |
| Mobile Responsiveness: | We ensure the application is mobile-friendly to accommodate users accessing it on various devices.  Prioritize a responsive design that adapts to different screen sizes. | React.js ,CSS3, Bootstrap /ReactNative. |  |
| User Profile Management: | Frontend (React):  react-chartjs: A library for creating interactive charts and graphs, allowing you to visualize user data.  react-data-grid: A library for displaying and managing tabular data, such as saved properties and recent searches.  react-recommendation-engine: A library for building recommendation systems, allowing you to suggest relevant properties based on user preferences and behavior.  (Backend-Node.js):  express-session: A middleware for managing user sessions, enabling you to store and retrieve personalized data for each user.  mongoose-user-plugin: A plugin for Mongoose, providing additional methods for managing user data, such as saved properties and recent searches. |  |  |
| Transparent Property Information:  Display comprehensive and transparent information about each property, including high-quality images, floor plans, and key details. | Node.js:  realtor.com-api: A Node.js package for accessing real estate data from Realtor.com, one of the largest real estate websites in the United States.  zillow-api: A Node.js package for accessing real estate data from Zillow, another major real estate platform in the US.  trulia-api: A Node.js package for accessing real estate data from Trulia, a well-known real estate website in the US.  county-recorder-api: A Node.js package for accessing public records from county recorders' offices across the US.  tax-assessor-api: A Node.js package for accessing property tax information from tax assessor offices in various US jurisdictions.  React.js:  react-realtor: A React component library for integrating with the Realtor.com API, allowing you to display property listings and details on your web application.  react-zillow: A React component library for integrating with the Zillow API, providing components for displaying property maps, pricing trends, and other Zillow data.  react-trulia: A React component library for integrating with the Trulia API, enabling you to showcase property listings and neighborhood insights on your React application.  react-county-recorder: A React component library for interacting with county recorder APIs, allowing you to display property ownership records and other public documents.  react-tax-assessor: A React component library for integrating with tax assessor APIs, providing components for displaying assessed values, tax histories, and other tax-related information. |  |  |
| User Engagement Features:  Saved Searches: Allow users to save their property searches for easy access and notifications of new listings.  Favourites: Enable users to save their favorite properties for quick reference. | Frontend (React.js):  react-local-storage-hook: A simple and lightweight hook for managing data persistence in local storage, ideal for storing saved searches and favorites.  react-context-api: A built-in React API for providing global state management, allowing you to share saved searches and favorites across components.  react-redux: A popular state management library for React, offering a more structured approach to managing user data, including saved searches and favorites.  Backend (Node.js):  mongoose: A popular object-relational mapper (ORM) for Node.js, enabling you to store user data, including saved searches and favorites, in a database.  express-session: A middleware for managing user sessions, allowing you to associate saved searches and favorites with specific users.  socket.io: A real-time communication library for Node.js, enabling you to send real-time notifications to users when new listings matching their saved searches are found.  Example Implementation:  Saved Searches:  Frontend: Use react-local-storage-hook or react-context-api to store saved searches in local storage or a global state management context.  Backend: Implement a database schema using mongoose to store saved searches for each user.  Favorites:  Frontend: Use react-local-storage-hook or react-context-api to store favorited properties in local storage or a global state management context.  Backend: Implement a database schema using mongoose to store favorited properties for each user. |  |  |
|  |  |  |  |
|  |  |  |  |

List of developer packages(Raddaf)