BEE END-TERM
Documentation

Topic - Real Estate Website

Submitted by:
Nandini Sharma
UID - 2010991951
Group - 23

Submitted to:
Mr.Anupinder

Abstract

The purpose of the real estate project is to create a platform for users to easily search and filter properties based on various criteria such as number of bedrooms, bathrooms, price and whether they are looking for to buy or rent. The frontend of the platform is developed using React, backend uses SQL database and Node.js API(along with Axios for connecting frontend API calls). The user interface is designed to be user-friendly and visually appealing, allowing users for an efficient and enjoyable property search experience. The project aims at streamlining the process of searching for a property, making it easier for users to find their dream home or investment property.

Acknowledgement

First and foremost, I would like to thank Mr. Anupinder who
guided us in doing this project. He provided us with invaluable advice. His motivation and help contributed tremendously to the successful completion of the project.
Also, I would like to thank my family members and friends for their support.
Without their support, I couldn’t have succeeded in completing this project.

Introduction

ElixirBricks, name for the real estate website, is a one-stop for searching one’s dream home, with filters to search a property with comfort and ease. The platform allows user to add and manage their own properties, search for properties and calculate estimated monthly payment using the loan calculator. The website provides an easy to use interface for users to list and find properties, as well as access information on property listings. With its comprehensive features and functionalities, this real estate website is the perfect solution for anyone looking to buy, sell or manage properties.

Technologies used

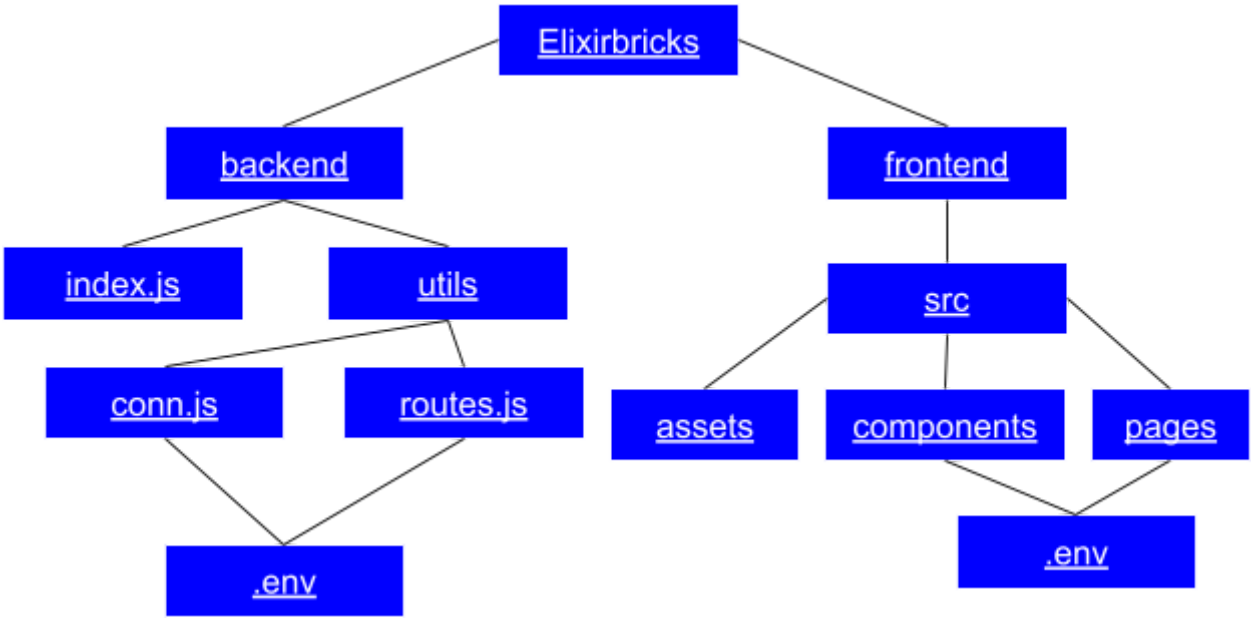
Following are the technologies used in project:

- Frontend
  - React - creation of UI
  - Vanilla CSS - styling of webpages
  - Fontawesome - usage of icons

- Backend
  - MySQL (CLI or phpMyAdmin(GUI-based)) - relational database
  - Express with Node.js - for building the RESTful API
  - Axios - for connecting the frontend to the backend

- Other libraries used
  - Body-parser - parsing middleware for accessing the content sent through frontend
  - Bcrypt - used for hashing passwords
  - Cors - for restricted allowance of resources on webpage
  - Dotenv - to access the environment variables
  - Cookie-parser
  - Express-session - to create time-based session for user

Directory structure



Components of project

The components for the project are:

- MainPage - choice for the user to login(existing user) or signup(for new user).

- LoginPage - Mainpage navigates to LoginPage if an existing user logins, fields required for login are username(unique) and password(hashed but not validated). Overall the form is validated for wrong/empty entries or an non-existing user.

- SignupPage - for new users, user has to give in their email, username and password for registration to website. The form is validated for wrong/empty entries. After signup, user is redirected to loginPage for logging into the website.

- Home - This is the entrypoint for a non-admin user with username displayed in the navbar and complete authentication and authorization functionality. UI is simple to use and easy to understand.

- Properties - This is the heart of the website where one can filter properties and fetch results accordingly. Filtering is done based on number of bedrooms, bathrooms, buy/sell property, area, price(range from minprice to maxprice).
- AddProperty - For a loggedin user, one can add her property using a basic styled form with fields validated, and a submit button, which stores the entered property mapped to the respective user account.

- EMICalculator - It calculates emi based on principal, interest and tenure entered by the user, using the sliders given. Also a

doughnut chart shows the principal to interested amount ratio.

Database schema

Database name:
realEstate

- users
- properties

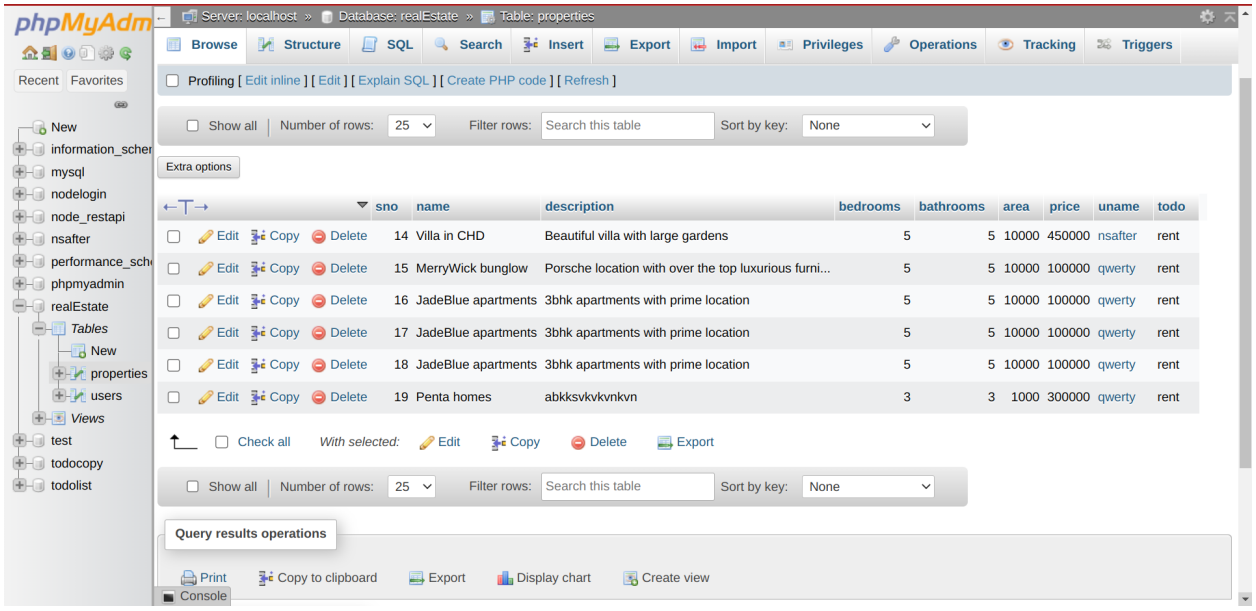
Following are the tables:

Users:

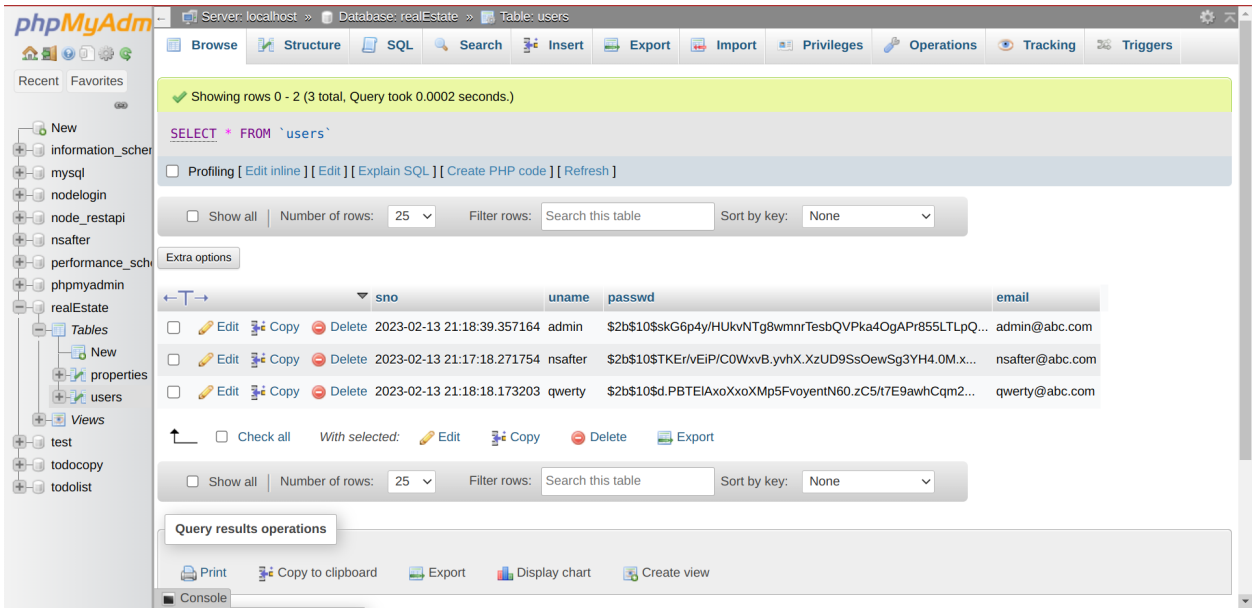
Sno.	Column_name	Data_type	Constraints
1	sno	timestamp	Not null
2	uname	varchar(255)	Primary key
3	passwd	varchar(255)	Not null
4	email	varchar(50)	Not null

Properties

Sno	Column_name	Data_type	Constraints
1	sno	int	Primary key Auto increment
2	name	varchar(20)	Not null
3	description	varchar(255)	Not null
4	bedrooms	int	Not null
5	bathrooms	int	Not null
6	area	int	Not null
7	price	int	Not null
8	uname	varchar(255)	Foreign key references users(uname)
9	todo	varchar(5)	Not null



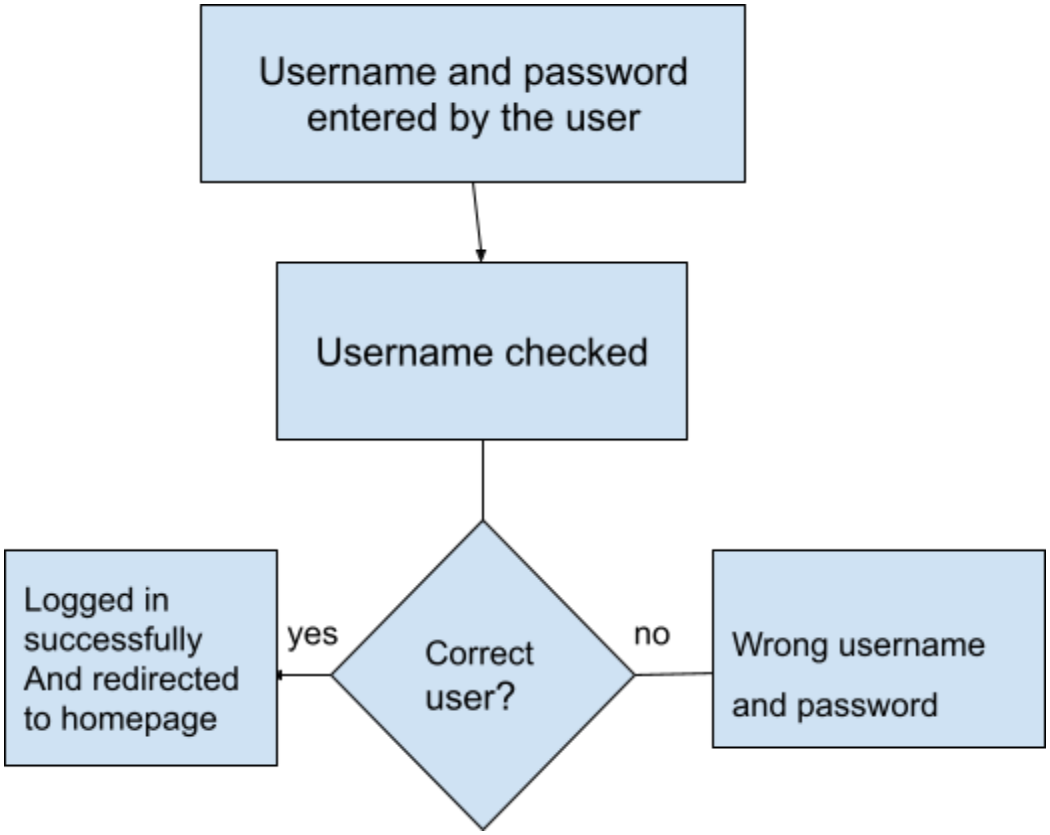
Properties table of realEstate database



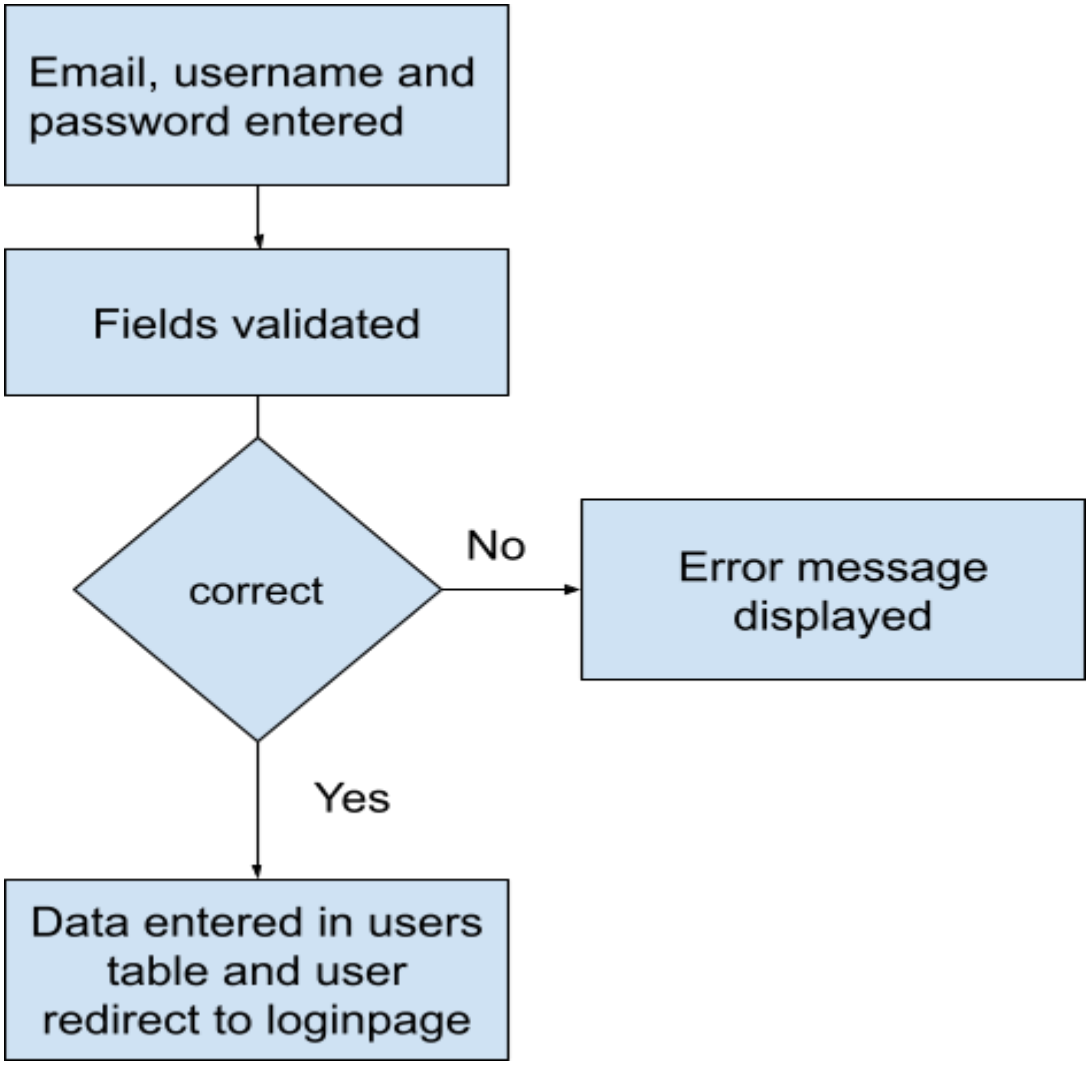
Users table of RealEstate database

Data flow

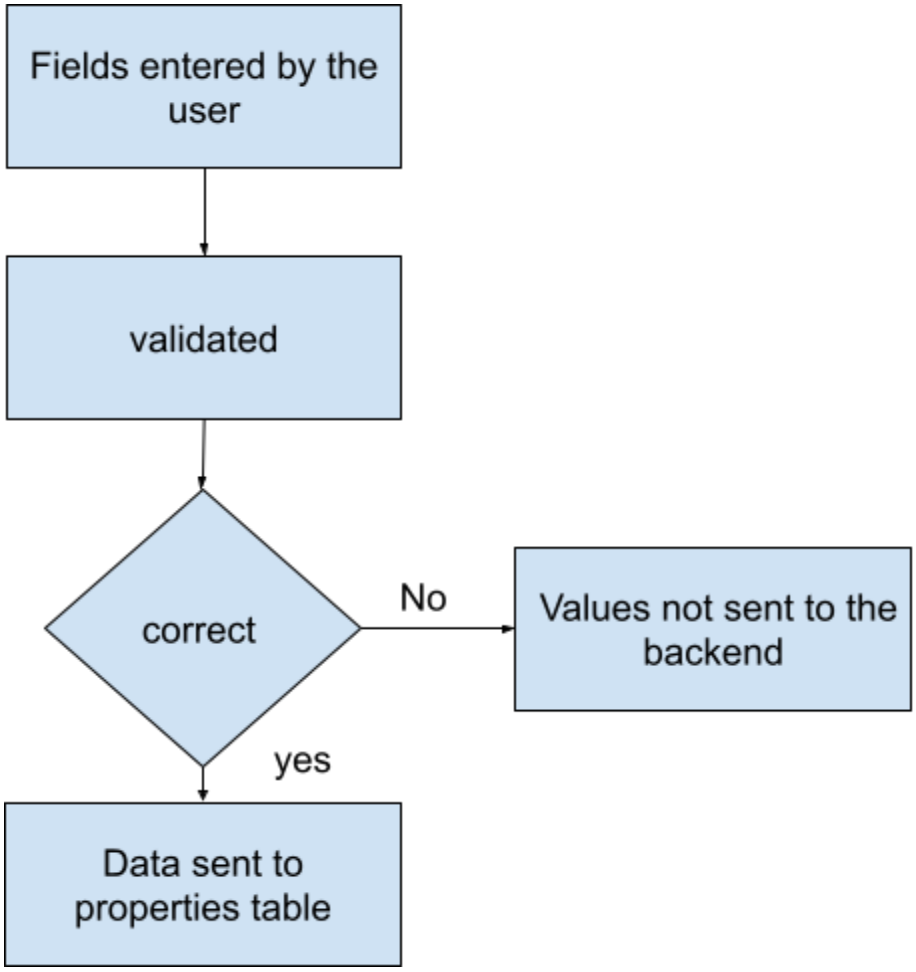
LoginPage



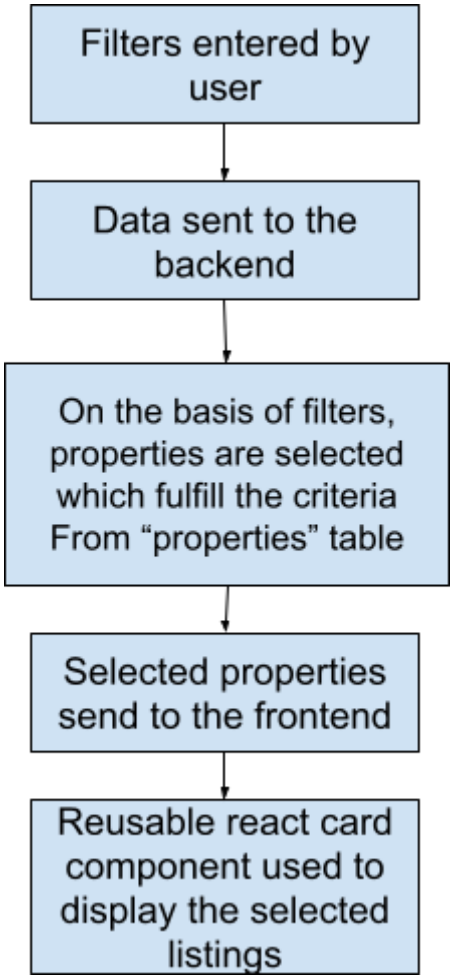
Signup


Email, username and password entered
Fields validated
correct
No
Error message displayed
Yes
Data entered in users table and user redirect to loginpage

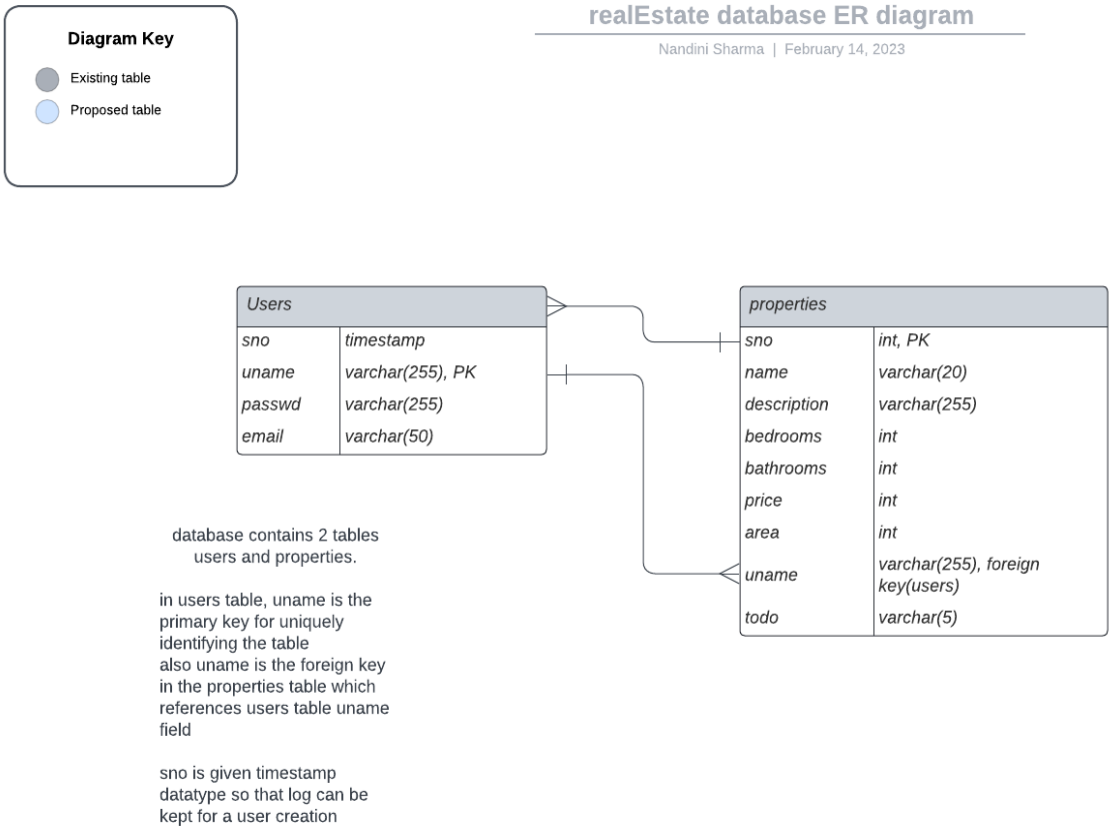
PostProperty



Property listings



Database ER diagram

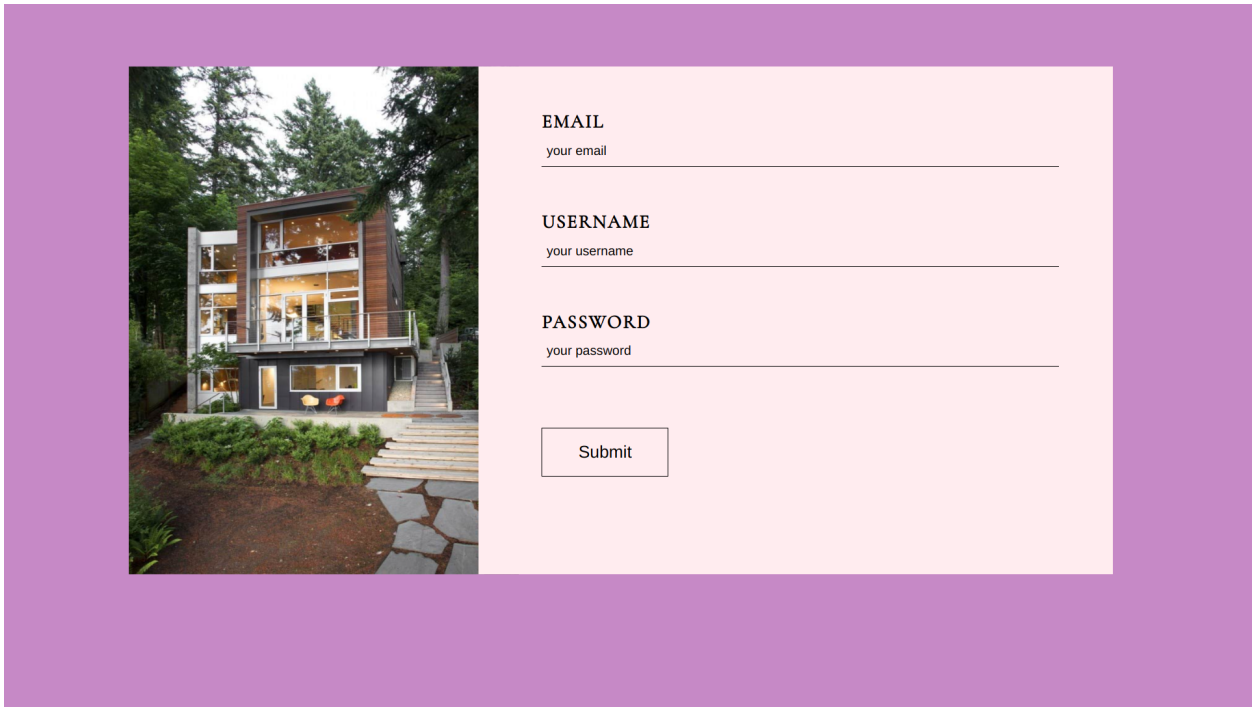


Data flow and navigation of project

The main page of Elixibricks where authentication and authorization of users is done

The main page of the website provides 2 buttons, login and signin
Login - redirects to login page for existing user
Signup - redirects new user to register

Register page:



Registration page for new users

2 types of users allowed

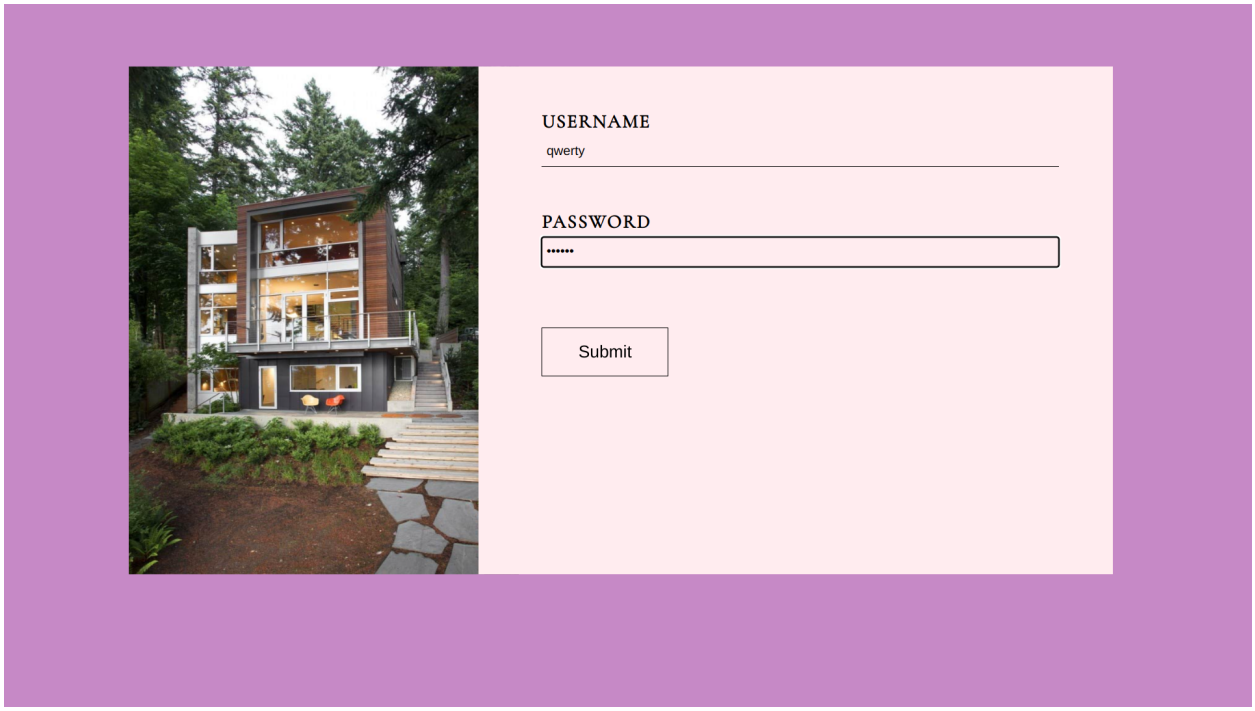
- General user - requires to fill the emailid, username and password, here username must be unique. Password is hashed at the backend for maintaining user security
- Admin - maintains users and property listings

Flow

Data entered by user -> useState hook used for the values entered by user -> validation performed -> if correct data sent to "/register" api(POST) -> password hashed with 10 saltrounds -> data inserted to "users" table using insert query

```
INSERT into users(uname,passwd,email) values(?,?,?)
```

Login page:



Login page

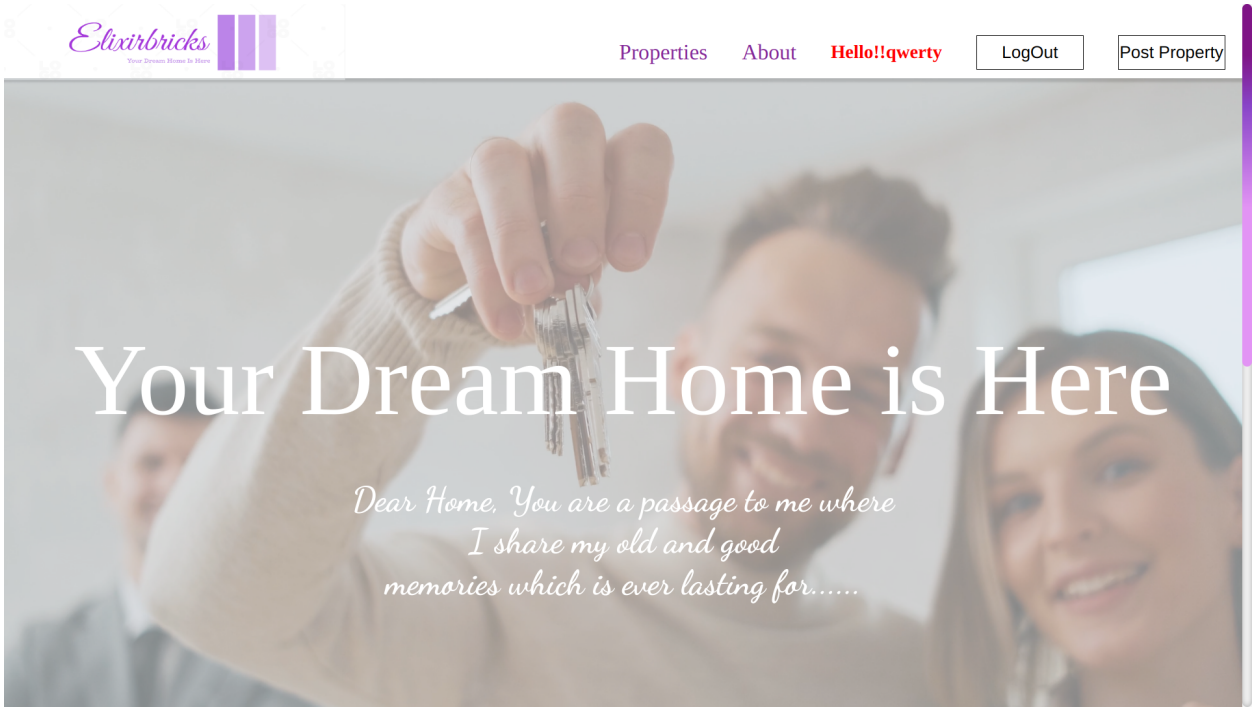
Flow

Data entered by user -> useState hook used for the values entered by user -> validation performed -> if correct data sent to "/login" api(POST) -> username searched in "users" table -> match found user entered password compared with hashed version -> correct ? session created -> user redirected to homepage -> else error message {"wrong username and password!!! "}

If user not found -> error message {"user does not exist "}

```
select * from users where uname=?
```

Home page



Home page of Elixibricks

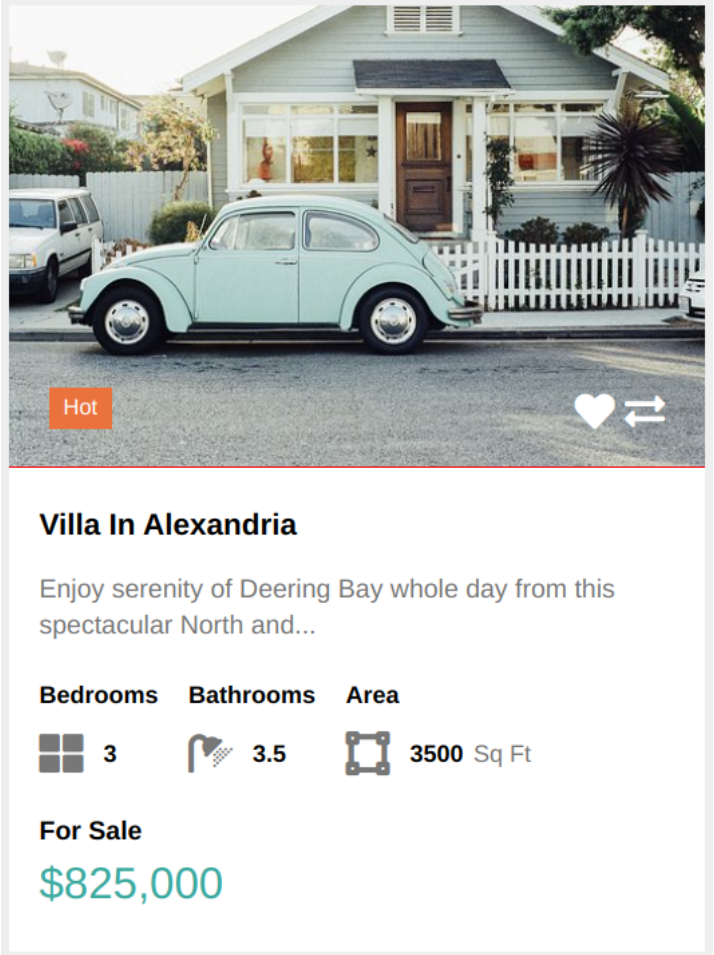
Flow

Current session user name mentioned in the navbar -> username fetched through “/logincheck” api(GET)-> if session exists, username returned

For logout button in Navbar -> when clicked -> api call to “/logout”(GET) -> if session exists, following operation is performed.

```
if (req.session.user) {
   req.session.destroy();
   res.clearCookie("userid");
   res.send({ message: "successful logout" });
 }
```

Property card

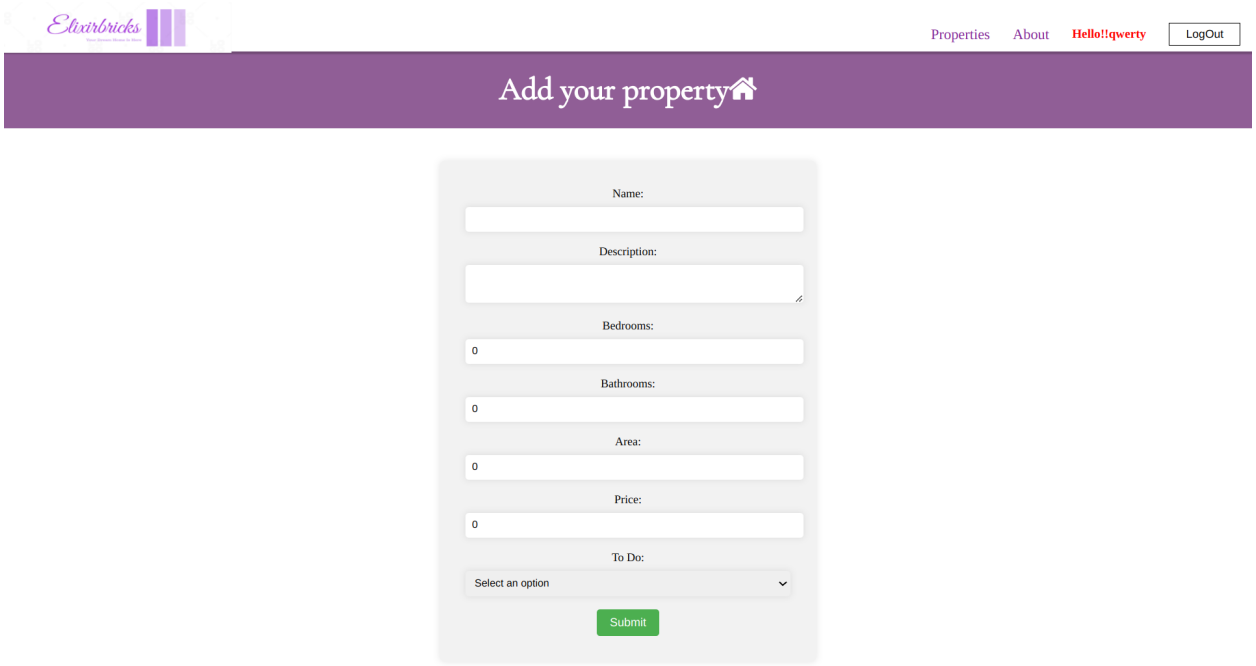


Data requested from “properties” table “/api/getdata ”(GET) -> data passed to card component through props

```
<Card
              url={img2}
              name={item.name}
              description={item.description}
              bedrooms={item.bedrooms}
              bathrooms={item.bathrooms}
              area={item.area}
              price={item.price}
              todo={item.todo}
            />
```

Card component with data passed as props

Addproperty



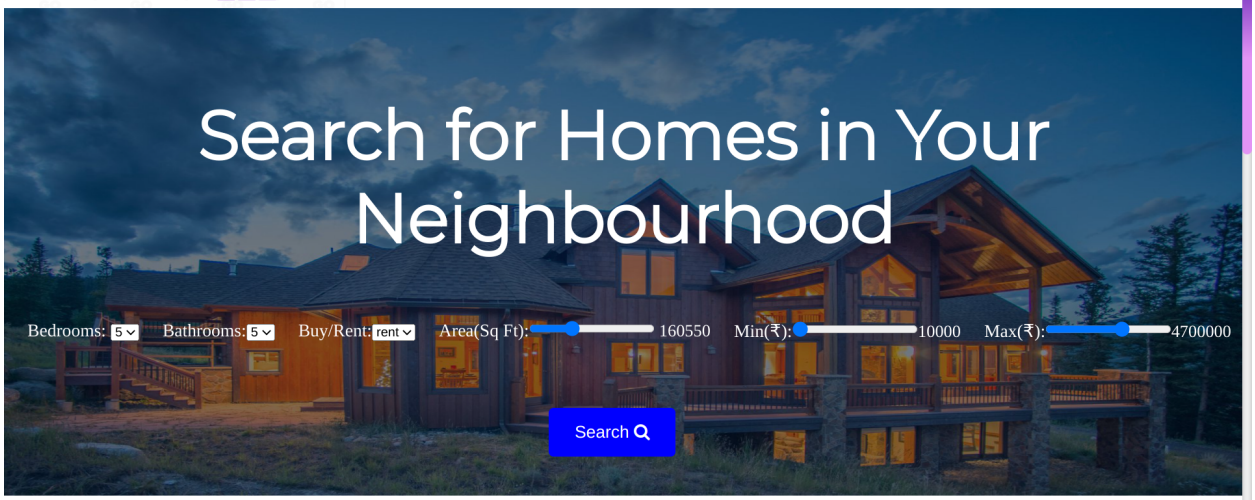
Form for entering a new property in "properties table"

Data entered by user -> useState hook used for storing the values -> data sent to "/addproperty"(POST) -> data inserted to "properties" table

```
INSERT into
properties(name,description,bedrooms,bathrooms,area,price,uname,todo)
values(?,?,?,?,?,?,?,?)
```

Elixirbricks
Your Dream Home Is Here

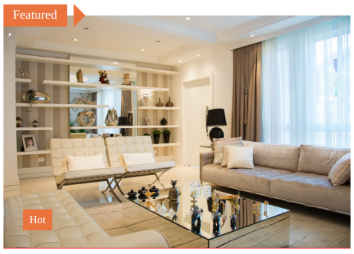
Properties About Hello!!nsafter LogOut Post Property



Not sure about your budget.. check our emi calculator

Filters entered by the user

Not sure about your budget.. check our emi calculator



Villa in CHD

Beautiful villa with large gardens

Bedrooms	Bathrooms	Area
5	5	10000 Sq Ft

For rent

₹450000

MerryWick bunglow

Porsche location with over the top luxurious furnishing

Bedrooms	Bathrooms	Area
5	5	10000 Sq Ft

For rent

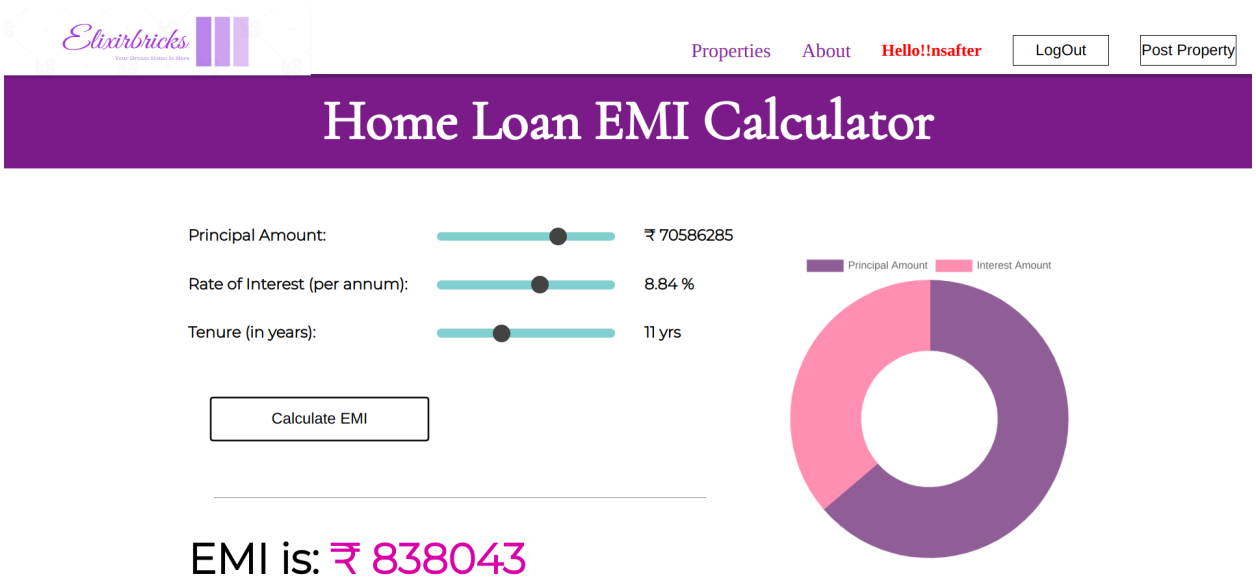
₹100000

Results on the basis of filters

Flow

Filters entered by the user -> data sent to "/api/filterdata"(POST) -> response fetched -> if length of response is 0 ? message shown {"Oops!! No match found!!!"} else -> data shown in form of card components

EMI calculator



calculates the emi on a principal value, for a particular timespan and rate of interest(anually)

Formula used:

P x R x (1+R)^N / [(1+R)^N-1] where-

P = Principal loan amount

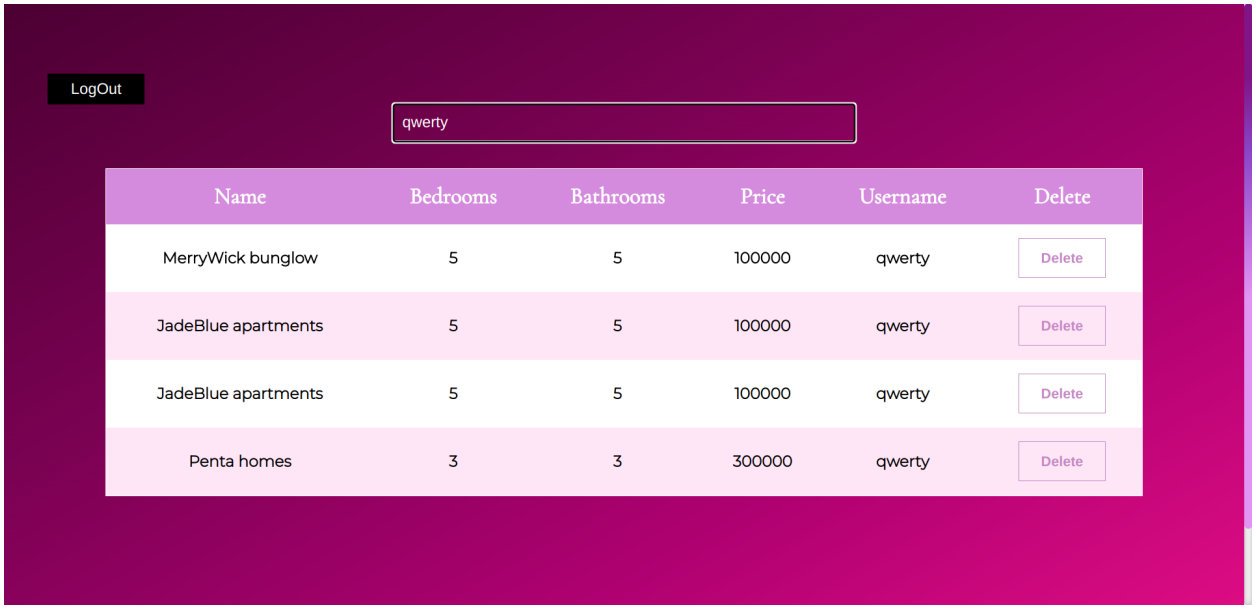
N = Loan tenure in months

R = Monthly interest rate

The rate of interest **(R)** on your loan is calculated per month.

R = Annual Rate of interest/12/100

Admin page



Admin page

Flow

On page load, data fetched from "properties" from "/api/getdata" (GET) -> received data displayed in form of table

Search bar - searching on basis of name of property and username.

Deleting a property -> id of specific property sent to "/propdel/${id}" (DELETE) -> data filtered on basis of sno field of "properties" table, if match found -> data deleted and page refreshed.

Conclusion

In conclusion, the real estate project “Elixirbricks” provides a comprehensive solution for users to search and filter properties with ease. The user-friendly interface and integration of various technologies make the platform efficient and reliable. The ability to add new properties to the database providers a convenient platform for real estate agents to showcase their properties. The project was successfully developed and tested, and it is ready for deployment. The developer is confident that this platform will meet the needs of users looking to buy or rent properties and enjoyable experience. It is a valuable addition to market and developer is proud to have been a part of its development.