Enabling a Safe and Immersive Online Home Search Experience in Kenya

EasyHomes.Ke

About the Project **3**
  Features 3
Discussion of Literature **4**
Market Research **5**
  SWOT Analysis 6
    Strengths 7
    Weaknesses 7
    Opportunities 7
    Threats 7
  PESTEL Analysis 8
User Guide **9**
  General 9
    Types of Users 9
    Sign up/ Login 10
    How to access main bar links on mobile? 11
    Filters 12
    Viewing a property 12
    Posting Properties 14
    Editing Properties 15
    Deleting Properties 16
  Admin specific instructions 17
    Verify Users 17
    View Reports 17
    How to create more admin roles? 18
User Interface / User Experience Design and Testing **19**
  Evaluation 23
Developer Documentation **25**
  Root Folder structure 25
  Front-end 25
  Backend 27
  Backend-frontend integration 28

Entity relationship diagram 30
CRUD user rights 31
Error handling 32
Form handling 33
User session management 35
Running the project locally 37
Pre-requisites 37
Running the application 38
Accessing the application 39

Project management methodology 40
Team composition and roles 42
Weekly Meeting Minutes and Sprint Reviews 43
Collaboration using GitHub 45

Automated testing 46
Backend 46
Frontend 46

User testing 50

Iterative design 52
Design phase 52
Development phase 52
Testing phase 53

Critical evaluation and discussion 55
Challenges 55
Our performance 55
Major flaws 56

Future development 57

Appendix 1: Iterative design screenshots 58
Properties search filter 58
View property page 59
Unauthorized page 60
Reports page (admin role) 61

Bibliography 62

About the Project

Moving or purchasing a house can be a stressful and troublesome experience, but in Kenya, it can also be a risky one. This is because one can be scammed while looking for a home to both buy or rent. While many people still opt to search for property the traditional way i.e. through a real estate agent, new digital solutions have started to emerge for people who prefer to explore more options and avoid the steep fees of property agents.

Kenya is particularly prone to property hunters falling victim to scam artists who take advantage of an unregulated market (Abeti 2017, Mwangi 2020). From our own user research and questionnaires, we estimated that on average 50% of the people trying to buy property in Kenya have fallen victim to some form of fraud.

Additionally, there are a handful of property applications in Kenya that offer viewings of property listings. However, they do not protect people from con artists or provide an immersive house hunting experience.

With all this in mind, we have created EasyHomes.ke, an easy-to-use modern web application that is trustworthy because it is designed to enable a safe, smooth, and immersive property searching experience for the Kenyan market but can also be rolled out internationally.

For testing and evaluation, the project is currently deployed at:

https://cm2020-agile-software-projects.vercel.app/

Features

From our extensive research and feedback from potential users, we decided to implement the following features into the application.

We understand the importance of user data and so we decided to have user feedback for the verification of each listing. We also designed certain features to decrease the chances of fraud. The features implemented include:

1. A state-of-the-art modern, dynamic web application that runs on Node.js and is built with Loopback 4 and Next.js. The web application allows users to browse through property listings and apply a range of search filters to narrow down to the specific type of property they are looking for.
2. Only allowing authenticated i.e. logged-in users, to create property listings. They need to have an account and then log in via their email and password. The authentication is based on the state-of-the-art JSON Web Token (JWT), an encrypted string containing user profiles such as email, user id, and other features.
3. Any authenticated user can post properties, however, administrators of the website can set trusted users as verified. Verified users will have a green

"verified" badge on the properties they list. This, if used responsibly, will build trust with the users because they will be able to rely on the website's administrator's guarantee that those postings have been verified.

4. In the case where users suspect that a listing is fraudulent, spam, or otherwise inappropriate, they will be able to report such listings. The user must also be logged in to do that.
5. The reported properties can be removed by the administrator after confirmation. The users that list fraudulent properties can also be banned from the platform by administrators.

All these features are novel for the Kenyan market. Upon user research, through questionnaires, we confirmed that this will enable a substantial reduction in fraud within the realm of real estate transactions.

Discussion of Literature

The Kenyan real estate market has grown exponentially over the past two decades. In the year 2000, the real estate sector made up 10.5% of the country's GDP and in 2016, this had grown to 13.8% of the country's GDP (Cytonn.com 2021). The drivers for this growth have been:

1. Strong and stable growth in the economy driven partly by political stability as Kenya ushered in a new era of democratic rule
2. Infrastructural developments such as improved roads, utility connections, upgrade of key airports
3. Demographic trends such as rapid urbanization at 4.4% p.a against the world's 2.5% and population growth averaging at 2.6% p.a
4. Traditional affinity towards investing in real estate partly due to the lack of alternative places to invest due to underdeveloped capital markets

Residential real estate has witnessed the strongest demand with a nationwide housing deficit of at least 200,000 units per annum due to the growing middle class, general population growth, and rural to urban migration. The housing market is also primarily a rental market with affordability being the main challenge (Saif Properties, 2021). The Kenyan government has launched initiatives aimed at incentivizing developers to build affordable homes with the government aiming to build 500,000 new homes by the end of 2022.

The government established the Kenya Mortgage Refinancing Company (KMRC) so as to provide long-term mortgages at affordable rates. Further incentives include stamp duty tax exemption for first-time homebuyers and a 15% corporate tax rate relief for developers who build at least 100 affordable housing units a year. The National Housing Development Fund (NHDF) was also established and this acts as a saving trust for employees so that they can save to purchase a home in the future.

It is quite common for Kenyans to fall prey to unscrupulous property dealers posing as legitimate property owners or brokers during the process of buying property. Some of the reasons for this are:

1. The strong demand for real estate
2. It's an easy scam
3. There is little regulation to protect consumers
4. There is a lack of consumer watchdogs
5. Power is firmly in the hands of the owners of properties and not with consumers

Hundreds of fraud cases are reported every month (Safu 2020) with sophisticated scammers operating in broad daylight; attending property shows, advertising on billboards, and on television. As regulation is weak it is difficult for consumers to make the appropriate checks and perform their own due diligence.

Scammers exploit the demand for housing in Kenya, especially in the capital city of Nairobi which has a population of 4.4 million. There is a shortage of affordable housing units and scammers take advantage of the desperation (Mwangi 2020). There are also off-plan housing scams where potential homeowners are convinced to invest in homes that have not been developed yet however many have lost money in these schemes when the property developers fail to deliver.

A common con is when a scammer impersonates and purports to own a vacant home and shows it around to potential tenants - all the scammer needs is a key which is easily obtained especially if they also trick the homeowner that they are an agent. As rent in Kenya is typically paid three months in advance and rental deposit is also equivalent to three months rent. A new tenant will usually have to pay 6 months of rent before first moving into the house. Only after the victim pays and attempts to move in do they quickly realize that they have been conned and by that time there is little that they can do.

Market Research

The only competitors found that focuses solely on the Kenyan market were

- Buy Rent Kenya (https://www.buyrentkenya.com/).
- We also found another competitor, Property 24, (www.property24.co.ke) that offers properties in Kenya, but they have other websites for other countries e.g. https://www.property24.co.na/ for Namibia.

We found that other major competitors are not focused just on the Kenyan market, but across various markets, for example :

- Knight Frank.
- Rightmove.

In terms of the UI, the website design was very similar across competitors. The top bar of their landing pages included "for sale" and "for rent" drop-down menus and the options on the menu were to select the type of property for example, whether the user was after an apartment, house, or land. Buy Rent Kenya also has a "projects" tab which focuses only on the buying of new builds (properties currently still under construction). This "projects" tab does not seem to be offered by many other competitors. They also included an additional drop-down menu for "Advice" which includes, real estate news, décor & lifestyle, property guides, and neighbourhood guides.

There were also a number of similarities in design for the pages which displayed current property listings. For most competitors, above or to the right of the listed properties there would be a box for added filters such as the location, number of rooms, and the minimum/maximum price. Each listed property included the price or price range at the top, the estate agent's logo, and three individual buttons to contact the seller via different contact methods such as email, WhatsApp, telephone. Some competitors also gave the option to favorite/save a property so the user can refer back to it in the future. For "Buy Rent Kenya" they offer a unique page for "bedsitters", that is, property that only offers a single room in a house share. For other competitors, the bedsitter properties are listed together with other properties to rent, which can make searching for whole properties to rent slightly more difficult. In general, sites tend to all look very alike with similar layouts and offers.

One crucial disadvantage we have found is that for Buy Rent Kenya, when attempting to list a property there are very few security checks, and almost anyone can post a listing. This could in turn result in properties being fraudulently sold or rented, which through our user research, we have found to be an ongoing issue.

SWOT Analysis

The property listings industry is a competitive one. With so many different websites, apps, and brokers vying for attention, it can be difficult to find the right listing at the right price. That is why it's important to keep up with all of the latest trends in this ever-changing industry. We shall discuss challenges that we may face when developing a unique property listing application that prevents fraud: pricing pressure from other sellers, finding an agent who will work with your needs, navigating new technology like 3D renderings and virtual tours, and managing expectations during negotiations.

Property listings have become an integral part of the real estate industry. They are used by buyers and sellers to find information about homes for sale in their local area. The challenge with property listings is that they often involve a lot of work, without much reward.

A SWOT analysis was conducted as part of the initial market research in order to help identify the strengths and weaknesses of working in this chosen industry.

Description: A Property listing application that will be effective, enjoyable to use, and reduce the prevalence of fraud in the Kenyan market.

Strengths

- Technical knowledge
- Team members with professional experience
- Four individuals in the team (manpower)
- Digital payment methods widely adopted in Kenya
- The Kenyan market is very receptive to real estate products and investing

Weaknesses

- Lack of direct experience
- Insufficient domain knowledge (real estate)
- Low smartphone penetration rates
- No financing for the project
- Need to be more knowledgeable about legal and tax implications

Opportunities

- Unsaturated market
- Nairobi UN Headquarters (high salaried staff)
- The high population growth rate in Kenya
- Highly literate population - free education policy
- Big five agenda to provide affordable housing

Threats

- Existing competitors including traditional real estate dealers
- Poor Infrastructure
- Insufficient housing stock
- High levels of unemployment
- Disorganized urban landscape - prevalence of slums

PESTEL Analysis

A PESTEL analysis was performed and included below. Kenya is a developing country with an average annual GDP growth rate of around 3%. It has one of the highest population densities in Africa and its capital city Nairobi is experiencing rapid urbanization. The lack of proper infrastructure (electricity, water) and the high level of unemployment in Kenya is a central consideration as we build the application.

Political:

NGO'S, Nairobi as one of the UN Headquarters, Big Five Political Agenda

Economic:

The poor state of infrastructure, poor road networks, low level of employment, the poor state of transportation infrastructure

Social:

High population growth, youthful population (large child population), high literacy rates (good for data consumption), conservative and religious views, rural to urban migration, slums, free schooling provided by the government from primary to high school

Technological:

High literacy, good internet connection, smartphone penetration rates, cheap smartphones needed, 5G

Environmental:

Water scarcity, expensive energy, legal requirements for clean energy in new home builds, backup generation needed, power shortages

Legal:

Digital tax, environmental laws, property taxes need to be collected

User Guide

This section serves as a guide and explains how the application functions and how users can interact with various UI features inside the application. Users can be anyone that visits the site, the website supports admin roles, and normal user roles (either verified or unverified).

General

We have provided some dummy user login IDs for the purpose of testing the app and posting properties. You can also create your own account by following the guide.

```
User 1
Email:    "user1@example.com",
Password: "useruser"

User 2
Email:    "user2@example.com",
Password: "useruser"
```

Types of Users

General User (not signed in)

A user who is not signed into the application can still filter or browse the properties. However, for features such as posting properties, reporting properties, etc. the user needs to be a logged-in user.

Normal user (signed in & not verified)

Once signed up, a user will have a user account that will have an unverified role. Later they may request an admin to set their role to that of a verified user. Normal users as well as admins are able to post properties and edit the properties they have posted.
To view and edit the properties you have posted, please click on your username in the upper right-hand corner.

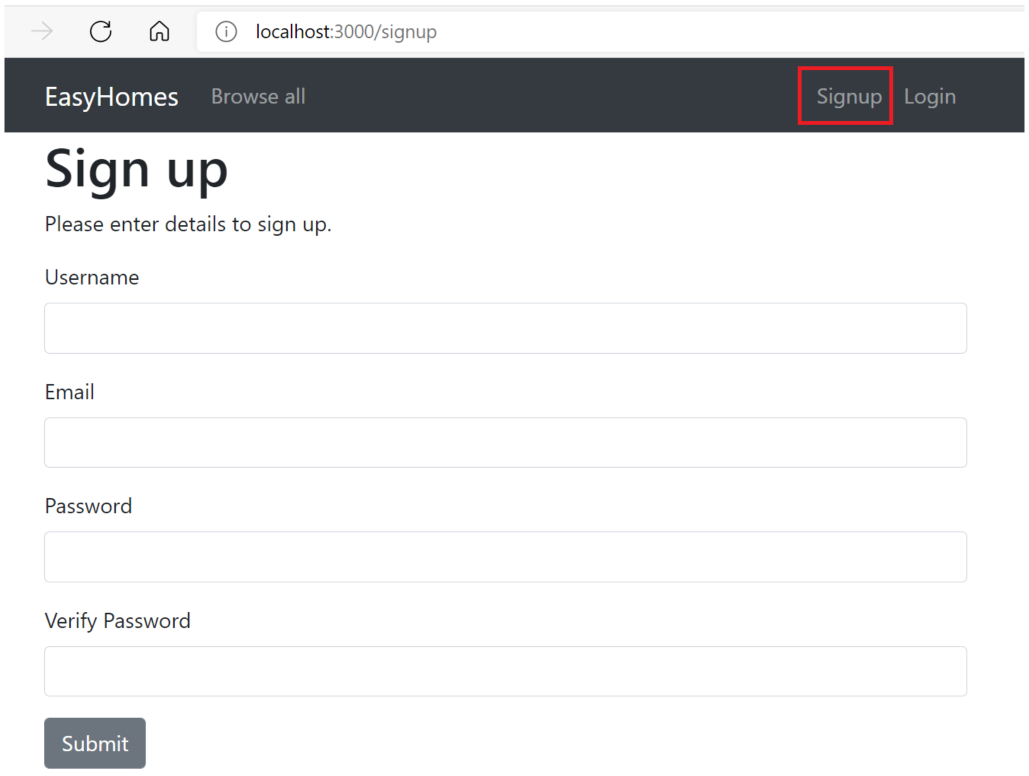
As a signed-in user, you are able to **report** properties that you believe are fraudulent or spam. To do this, you need to click the "View" button on a property card, and then on the following page, click on "Report". Remember that you are not able to report properties that you yourself posted, however you are able to edit those properties.

Normal user (signed in & Verified)
The verified users will get a green badge. This is visible when visiting the property they posted.

Description	
Posted by	user1 Verified

Sign up/ Login

- The Signup section can be reached from the “Signup” button on the navigation bar at the top.
- Users can sign up for the application using their email address and password.
- Please enter a username, email, and password. Password is verified to ensure you enter the exact password you think you are typing.
- A successful sign up will automatically redirect the user to the Home page



Sign up

Please enter details to sign up.

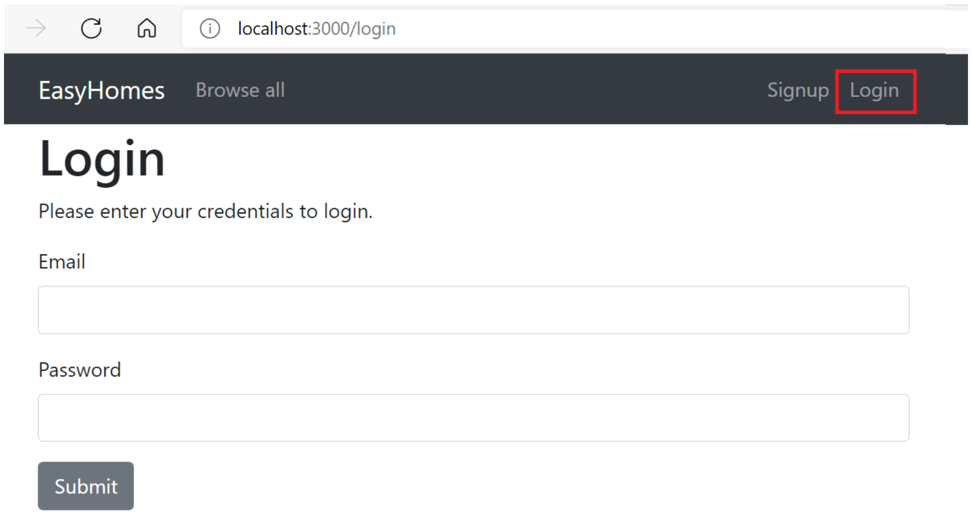
Username

Email

Password

Verify Password

Registered users can log in to their current account using the “Login” button on the navigation bar at the top.


localhost:3000/login

EasyHomes Browse all Signup Login

Login

Please enter your credentials to login.

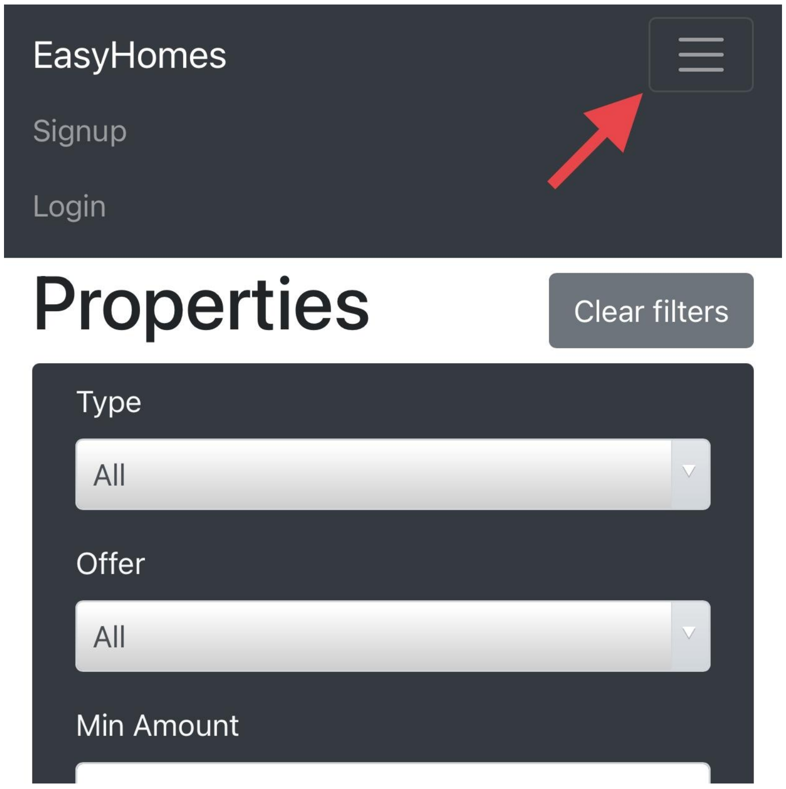
Email

Password

Submit

How to access main bar links on mobile?

To access main Navbar i.e. the top navigation bar links on mobile screen, please click on the hamburger menu indicated by the red arrow:


EasyHomes

Signup

Login

Properties

Clear filters

Type

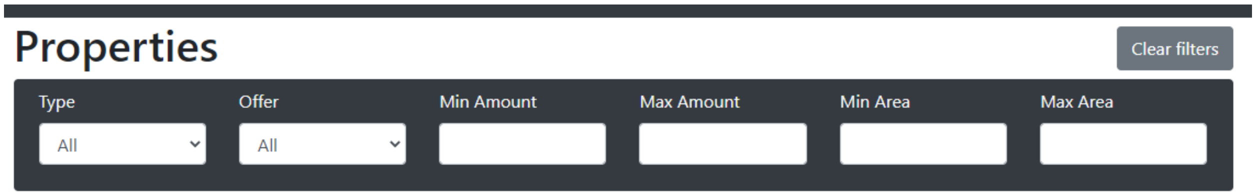
All

Offer

All

Min Amount

Filters



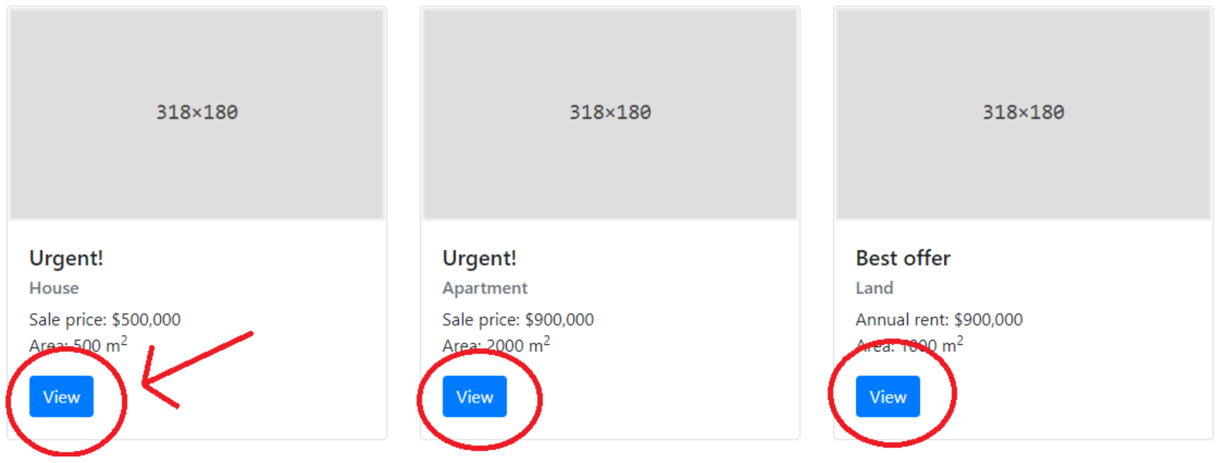
Users can filter different property listings through :

- Type - House, apartment, land commercial
- Offer - For sale, rent
- Price
- Area

They can also use the “Clear Filters” button to clear the filters they have entered and go back to the default page.

Viewing a property

Each respective property listing can be viewed by clicking the “view” button on each respective listing.



318×180

Urgent!
House
Sale price: $500,000
Area: 500 m^2
View

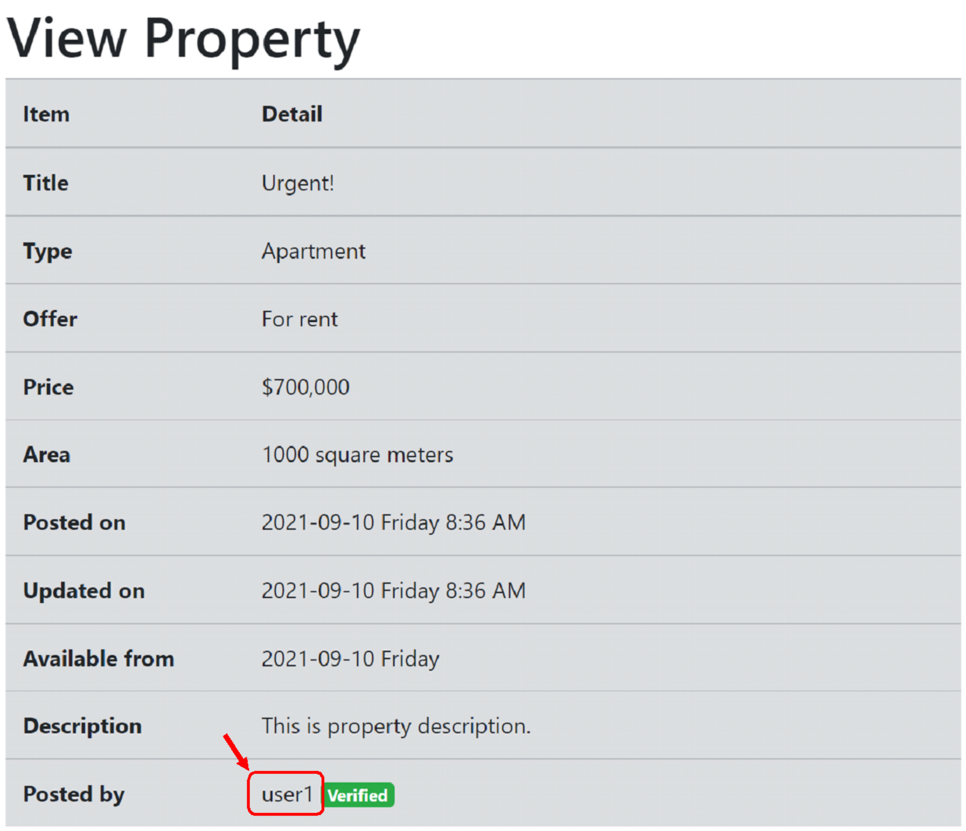
318×180

Urgent!
Apartment
Sale price: $900,000
Area: 2000 m^2
View

318×180

Best offer
Land
Annual rent: $900,000
Area: 1000 m^2
View

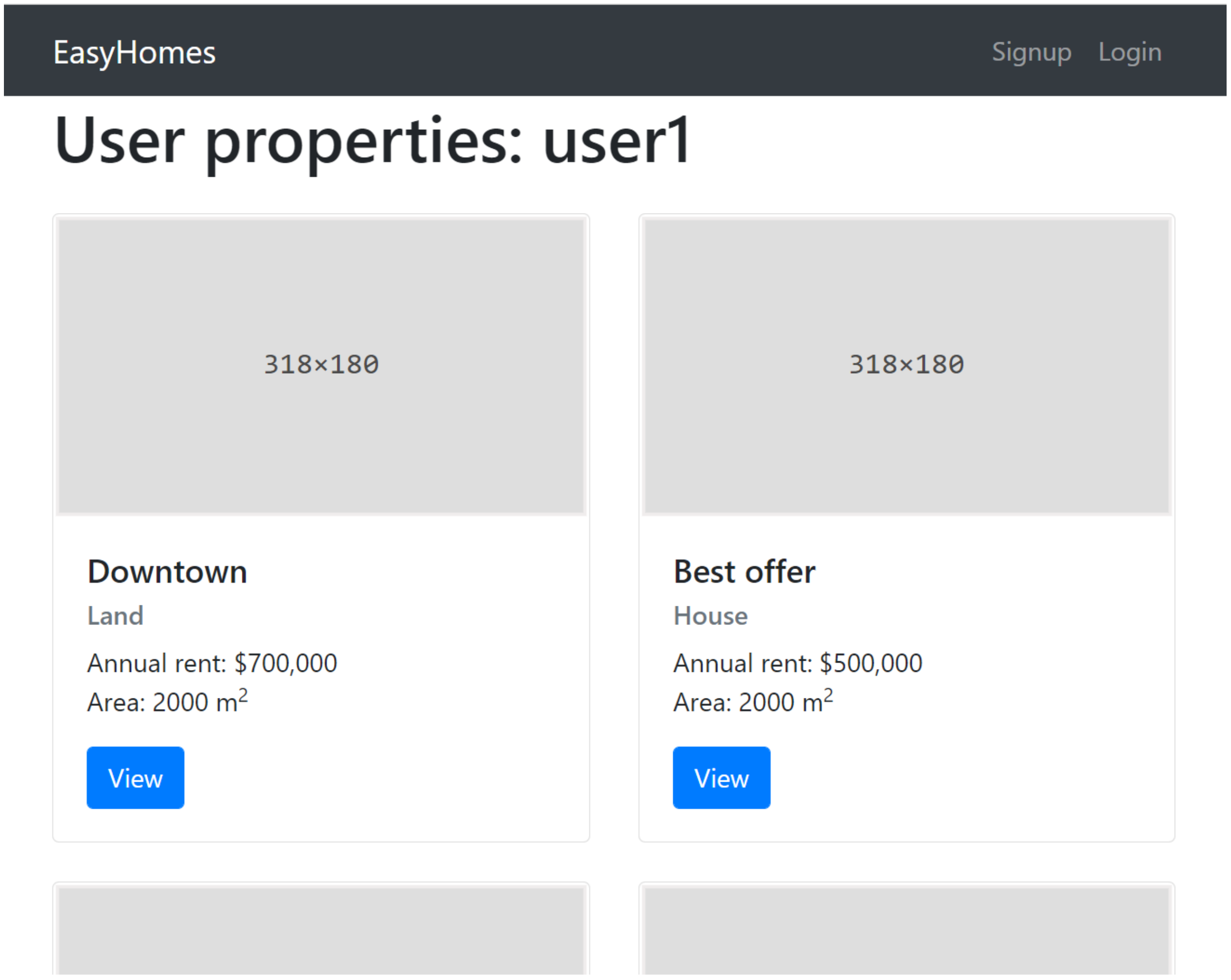
This will take the users to the view property page which shows the description of the corresponding listing that the user clicked on.



View Property

Item	Detail
Title	Urgent!
Type	Apartment
Offer	For rent
Price	$700,000
Area	1000 square meters
Posted on	2021-09-10 Friday 8:36 AM
Updated on	2021-09-10 Friday 8:36 AM
Available from	2021-09-10 Friday
Description	This is property description.
Posted by	user1 Verified

If you want to see all properties posted by a particular user, you can click their user name in the above screenshot which will lead you to a page that will list all properties listed by this user. The verified badge with a user's name indicates that this user has been verified by an admin.



User properties: user1

Downtown

Land

Annual rent: $700,000

Area: 2000 m^2

Best offer

House

Annual rent: $500,000

Area: 2000 m^2

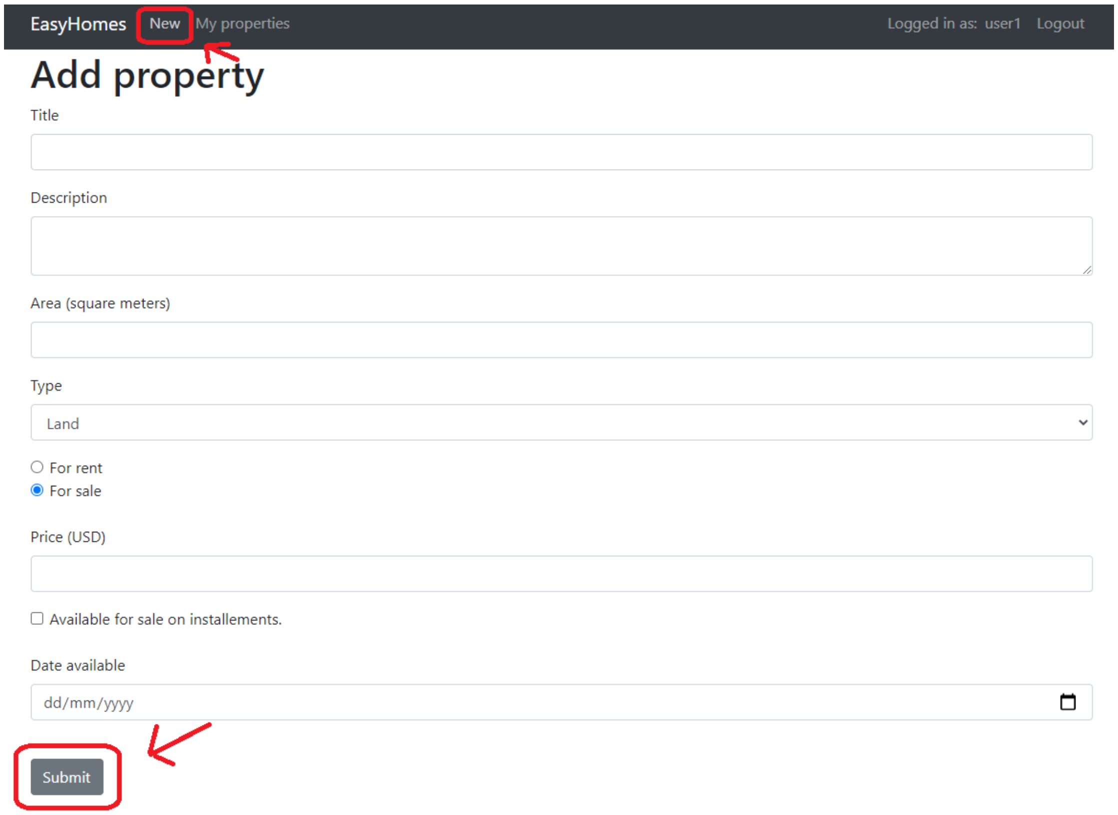
Posting Properties

Any logged-in user or an admin is able to post properties and edit the properties they have posted.

They can access this by clicking on the “new” button on the top left, which appears once they are logged in.

They can add various details to their posting such as:

- Title
- Description
- Area
- Type
- Price
- Date available


EasyHomes New My properties Logged in as: user1 Logout

Add property

Title

Description

Area (square meters)

Type

Land

○ For rent
● For sale

Price (USD)

☐ Available for sale on installements.

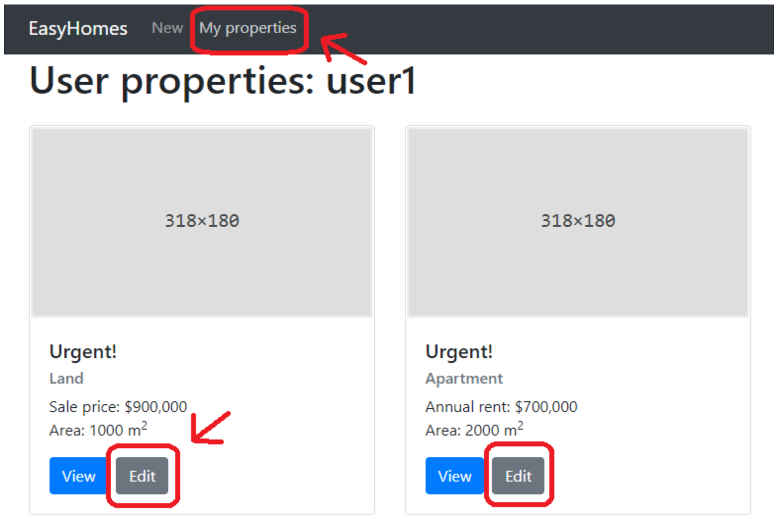
Date available

dd/mm/yyyy

Submit

Editing Properties

Users can only edit properties that they have posted.
They can access this by clicking on the “my properties” button on the top left, which appears once they are logged in. This gives them access to all the properties that they have posted


EasyHomes New My properties

User properties: user1

318×180

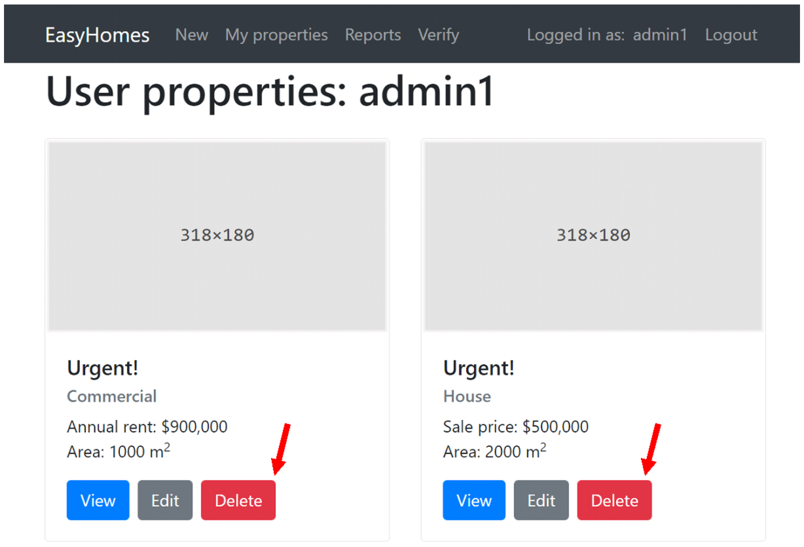
Urgent!
Land
Sale price: $900,000
Area: 1000 m^2
View Edit

318×180

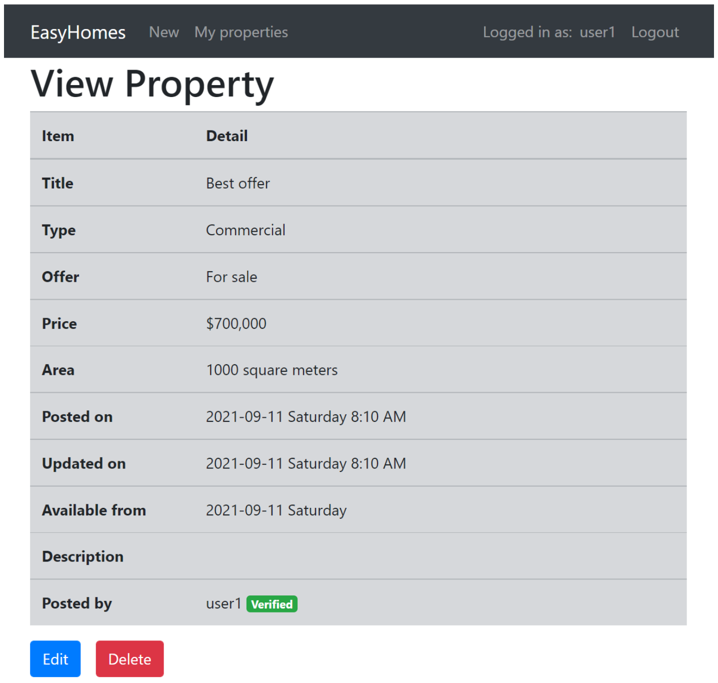
Urgent!
Apartment
Annual rent: $700,000
Area: 2000 m^2
View Edit

Deleting Properties

Users are also able to delete their own properties or properties that they have posted themselves. When signed in, they will be able to see the “Delete” button on property cards when they navigate to the “My properties” page.



To delete, simply click the delete button. You will then be prompted to confirm the delete action to ensure you do not accidentally delete a property. If confirmed, the property listing will be deleted. Property may also be deleted from the “View property” page as shown in the following screenshot.



Item	Detail
Title	Best offer
Type	Commercial
Offer	For sale
Price	$700,000
Area	1000 square meters
Posted on	2021-09-11 Saturday 8:10 AM
Updated on	2021-09-11 Saturday 8:10 AM
Available from	2021-09-11 Saturday
Description	
Posted by	user1 Verified

Admin specific instructions

The website also supports admin roles however, for security reasons, admin roles are not created using the signup form. For testing purposes, the following admin role is available for use:

Credentials

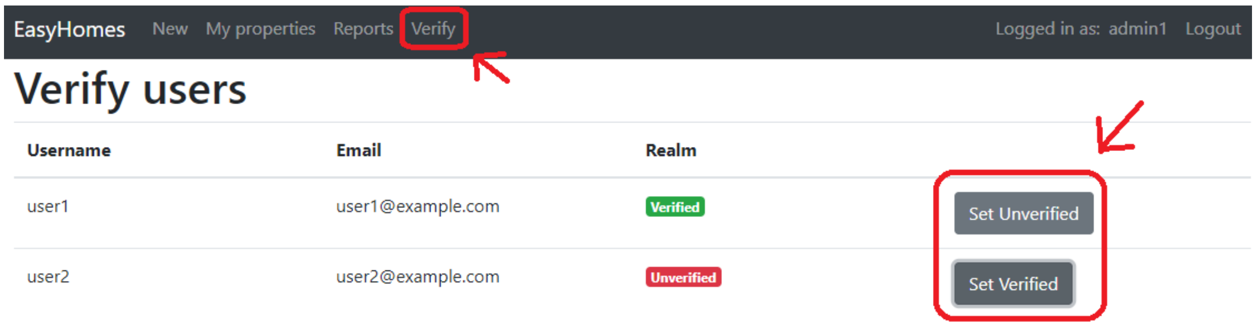
```
Email:    "admin1@example.com"
Password: "Admin!234"
```

Admin Users primarily have two privileges :

- Admins have the **right to delete properties** if users have reported them.
- Admins can also **set users as verified** or unverified.

Verify Users

To do this, click on "Verify" in the top navigation bar. Admins will see a list of all registered users and a button to set them as either verified or unverified.
Note: Admin roles themselves cannot be modified.

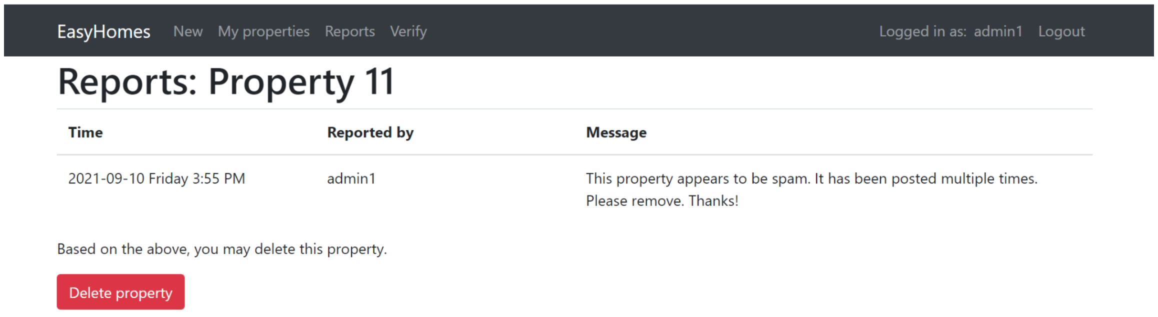


Verify users

Username	Email	Realm	
user1	user1@example.com	Verified	Set Unverified
user2	user2@example.com	Unverified	Set Verified

View Reports

Admin users are able to view and act on the reports filed by users on suspected fraudulent or spam properties. To do this, click "Reports" in the navbar on the top. If there are any reports, the administrator will see a table listing the properties and the number of reports against each property. Properties having more reports against them will be at the top. To view and act on the reports, click on "View" on any row and then you are able to see the actual message of the reports and you are able to delete the property in question if you are convinced that the reports are reasonable and legit.



Reports: Property 11

Time	Reported by	Message
2021-09-10 Friday 3:55 PM	admin1	This property appears to be spam. It has been posted multiple times. Please remove. Thanks!

Based on the above, you may delete this property.

How to create more admin roles?

Note: For security reasons, admin roles may only be created by a person having access to the source code which is usually the developer who deploys this application. There, the developers need to be requested to add new admin roles for you.

For developers:

Please go to /server/src/observers/seed.observer.ts and add a new object entry to the const users array. This can be used for creating other types of users as well, and for admin, the property realm should be admin. Please note that duplicate entries will fail, so double-check the values for email and username.

Once the above is done, please restart the server by stopping the server process and then type:

```
npm run start ## on /server directory
```

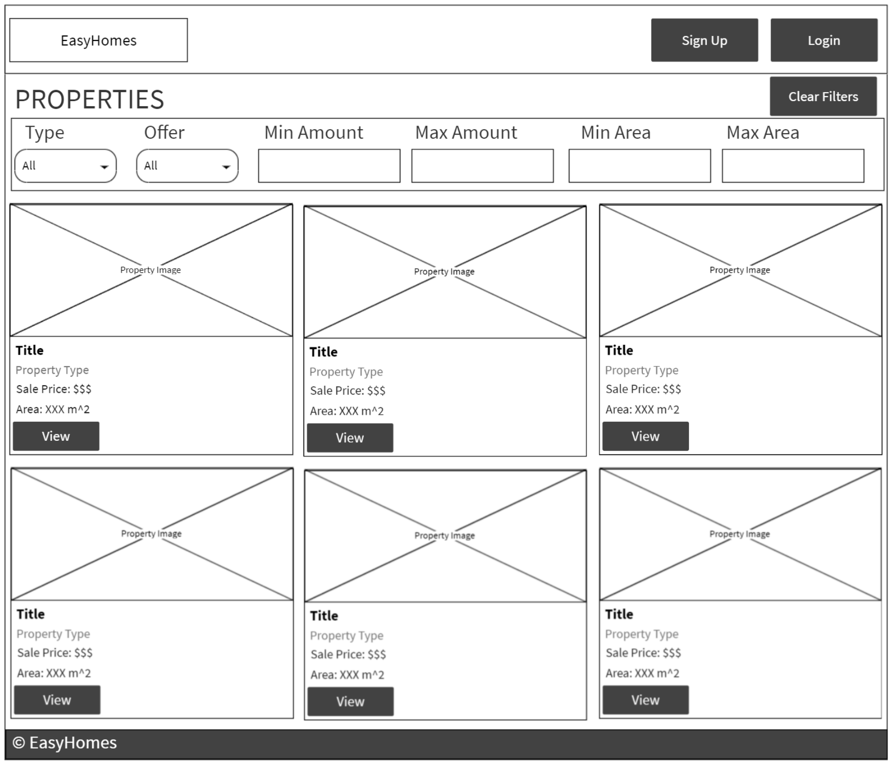
User Interface / User Experience Design and Testing

The design of the final high fidelity prototype improved upon the initial low fidelity designs and the functionalities needed for the application. These were used for the initial rounds of user testing. The challenge was to build features that cater to most of the users without overwhelming them

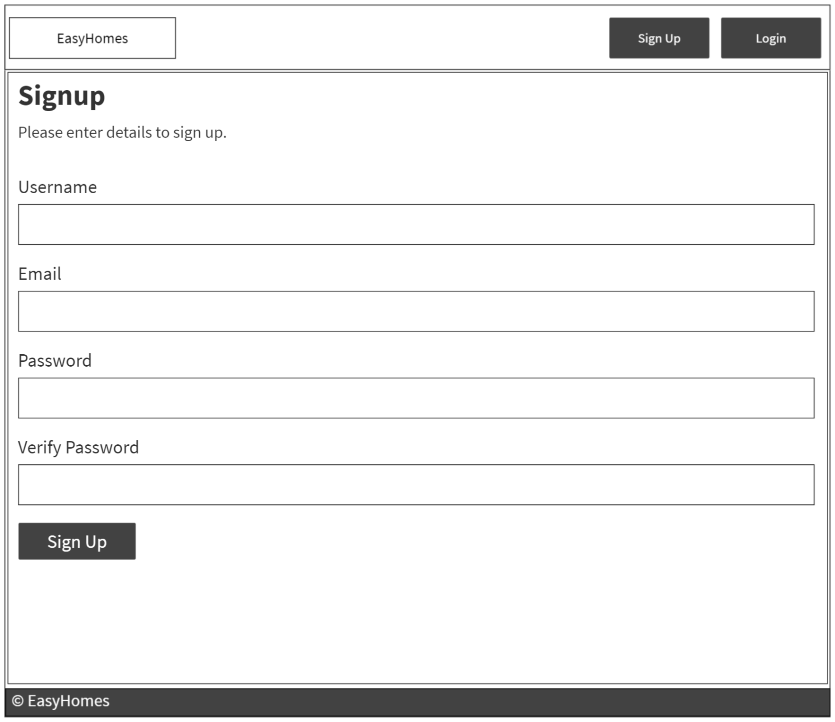
Below are the high fidelity prototypes of the application:

The home page contains basic search functionality with a call to action at the bottom allowing the user to view the details of the property.

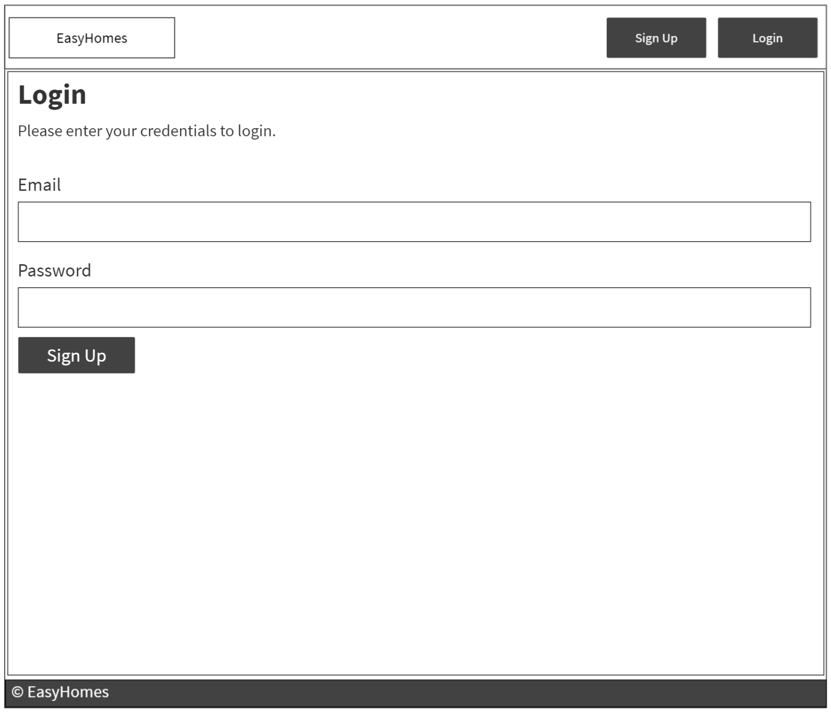
Home Page:



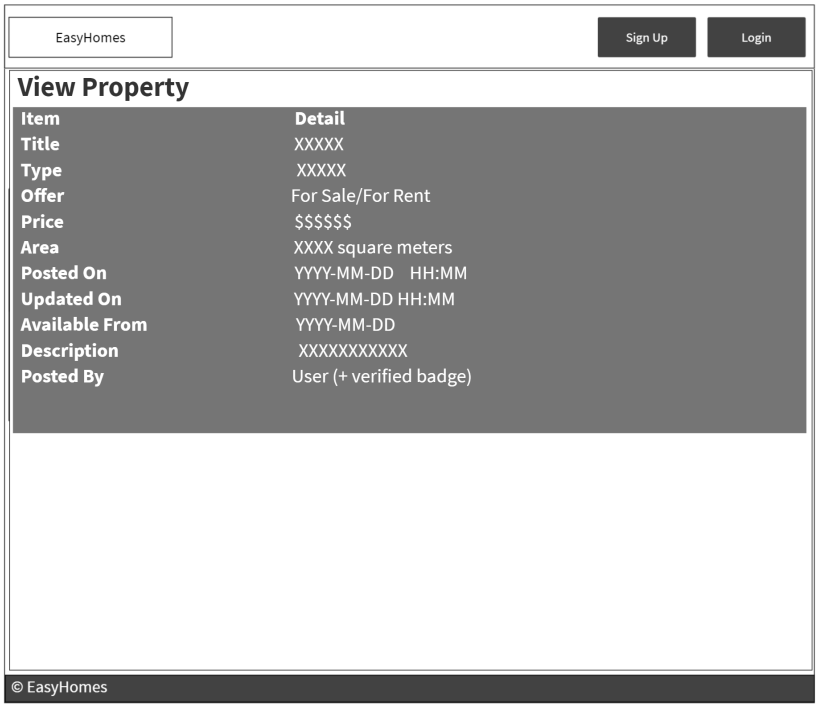
Signup Page:



Login Page:



Property View:



View Property

Item	Detail
Title	XXXXX
Type	XXXXX
Offer	For Sale/For Rent
Price	$$$$$$
Area	XXXX square meters
Posted On	YYYY-MM-DD HH:MM
Updated On	YYYY-MM-DD HH:MM
Available From	YYYY-MM-DD
Description	XXXXXXXXXXX
Posted By	User (+ verified badge)

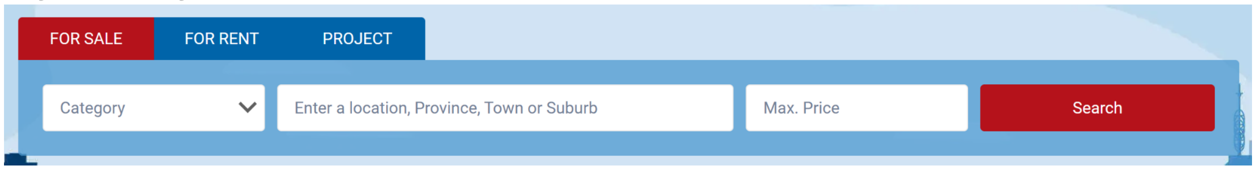
© EasyHomes

Based on initial research conducted on the largest competitors, we looked at what we felt these competitors did well, and anything we think could have been done differently, or improved upon. With these findings, we incorporated this into our design.

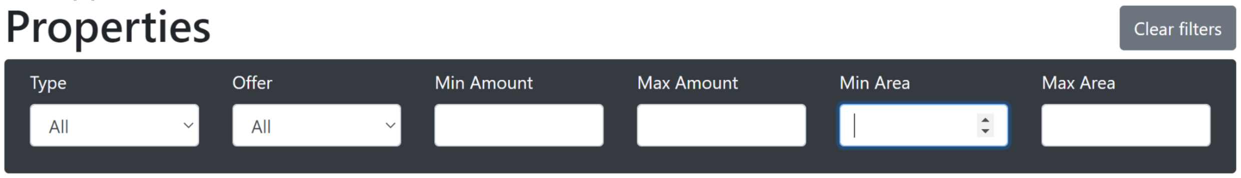
Looking at the website design of our primary competitor, BuyRentKenya, the top of the main search page included a horizontal set of search boxes/drop-down tabs to filter through properties. The filters were “Category” (i.e. the property type), “Location”, and “Max Price”. We decided to take on a similar approach as we found this simplified the process of property searching for users, and also noticed that a similar approach was adopted by a number of other competitors, e.g. Property24, Knight Frank, and Rightmove.

In our application, we included a set of property filters but felt more filters were necessary to fully simplify the property search process. From our own research, and speaking to users, we were able to agree on the six most relevant search filters. As well as the Property Type and Max Price filters, we also added additional filters for whether the property was for sale or for rent, as well as the minimum price, and maximum and minimum area.

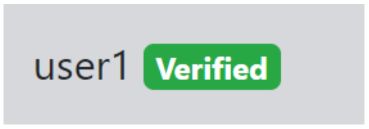
BuyRentKenya



Our Application

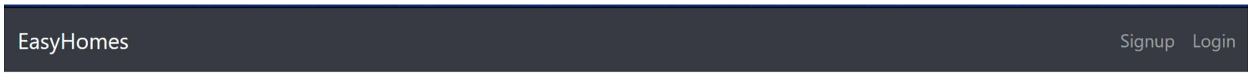


From previous research and readings, we found that property fraud was a common issue within the Kenyan market, and looked to see how our competitors tackled the issue, if at all. We found BuyRentKenya had very few tools to verify that users listing properties were not doing so fraudulently, and we felt in our design, our approach to tackling fraud should be made very clear. Website administrators can set trusted users as verified users, who will have a green "verified" badge on the properties they list. This aims to set our application apart from competitors as we are able to provide a greater guarantee for users that the properties they are viewing, and potentially interested in, have been listed by legitimate, trusted users.

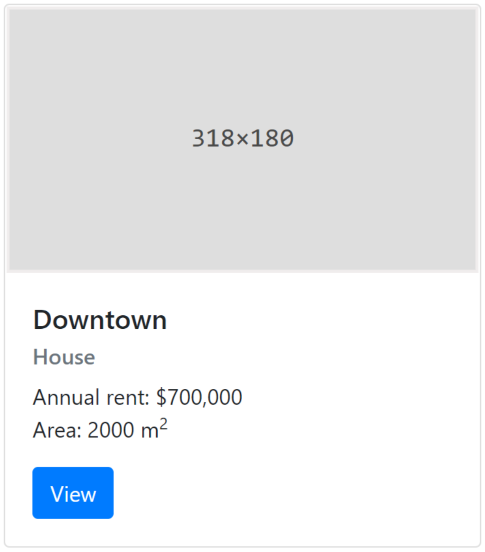


.

The login and signup buttons for users were placed at the top right of the screen for easy accessibility. This was a general trend that we found worked for our top competitors, and through user research, as it was commonly where users would expect to find any login/signup buttons. This placement increased the simplicity of the application and overall usability.



For property listings, a bold coloured "view" button was added underneath each property's details as it was found that users would generally prefer to read the initial basic information regarding the property first before deciding to view further, so the placement worked well. The bold colour also encouraged users to interact with the button.



Downtown

House

Annual rent: $700,000

Area: 2000 m^2

Evaluation

Throughout development, the team closely evaluated the progress made and the impact any added features or changes had on the overall usability and perception of the product. Our primary forms of evaluation were user questionnaires, A/B testing, and towards the end of development, heuristic evaluation (Sharp, Preece & Rogers 2019).

The team conducted regular user research, primarily in the form of questionnaires. These were conducted on a monthly basis with the aim of gathering feedback on the latest updates and additions made to the application.

For heuristic evaluation, we used a combination of Nielsen's (2020) heuristics as well as Budd's (2007) heuristics, as these were more specifically directed at website design and development. Using this set of heuristics, each team member took on individual tasks to complete, roleplaying as a typical user, and looked for any problems which may arise. Though heuristic evaluation may have its own disadvantages, such as issues being missed by researchers that would otherwise have been found when researching on real users, we found that given the limited access to users in-person, heuristic evaluation served as a strong alternative.

This coupled with user questionnaires and A/B testing, allowed for in-depth results to be gathered and implemented in our design, with the intention of creating a product with optimal usability.

The application was hosted in a cloud platform (Heroku) and the link, along with the form was sent to users for evaluation.

The evaluation was split into various categories for User roles and Administrator roles:

- Sign-up/login

- Posting a property
- Viewing a property
- Reporting a property
- Verifying a user (admin)
- Viewing user profiles (admin)
- View reported properties (admin)
- Deleting reported properties (admin)

The participants were told regarding the application and testing before they began :

“This is a property testing application meant for the Kenyan market.
The core premise of the application is to provide verification to legit properties and remove fraudulent ones.
You as a user can log in using the provided test profile or create your own profile by signing up.
Try posting a property. Try going to your user profile to see a property you have posted
Some users are verified by the admins, which specifies that the property posted by these users is trustworthy. So the general public can get an idea about when to use caution when viewing a new property.”

The team then reviewed the answers to questionnaires provided by the users.

A/B testing was done by having some users on video calls and making them test our different versions of the same functionality. The results were documented and the team decided on which feature was relevant during the sprint review sessions.

Developer Documentation

This section explains the technology used and the architecture involved in building this application. Front-end code is stored in the “client” folder. Back-end code is stored in the “server” folder.

Root Folder structure

Once cloned, please navigate into the repository folder. The repository actually contains two folders: “client” and “server”; the former being the frontend application and the latter being the backend API application. Both of these are written in JavaScript and require Node.js to run.

To separate the configuration files from the actual source code, both Loopback 4 and Next.js generate the boilerplate code with the actual source code of the application inside the “src” directory instead of the root of the application.

Front-end

The front-end of the application is built primarily using Next.js and TypeScript, although HTML, CSS also form the base of the application.

Next.js is a framework for production that is based on the React.js library. The team decided it is the ideal framework for this application based on its following advantages :

1. Server-side rendering of pages: Unlike most other popular frameworks, Next.js provides fully formed HTML pages along with the required fetched data to the client side. This allows for better Search Engine Optimisation (SEO) and faster performance. SEO is ideal for property applications, as it was evident from our market research that most users in Kenya search for properties using Google.
2. The folder-structure-based routing allows easier development and less code. Next.js under the hood creates pages for the TSX elements inside the “pages” folder.

Inside the “client” folder, the template seen is rendered by using the “yarn create next-app --typescript” command. This is a starter code to create a Next.js template along with its dependencies.

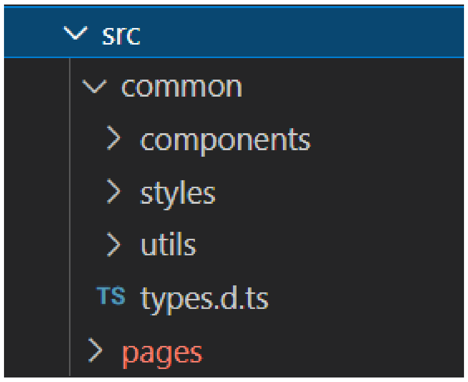
Next.js automatically creates “/path” for index pages inside the “pages” folder. There is no need for explicitly writing codes to route pages because Next.js scans the pages directory for JavaScript files that have default export of a React component. The folder and file names describe the URL that will be generated by the Next.js router.

Next.js is based on React.js so every element on the page is a TSX (typescript) component. These components are stored inside the “components” folder and imported for each corresponding page at the top.

TypeScript is a strongly-typed superset of JavaScript developed by Microsoft that compiles the written code into JavaScript. TypeScript reduces the need for writing extensive unit tests to check for types. TypeScript also enables “strict mode” by default while writing code which catches a lot of bugs during the compilation stage.

Page Structure

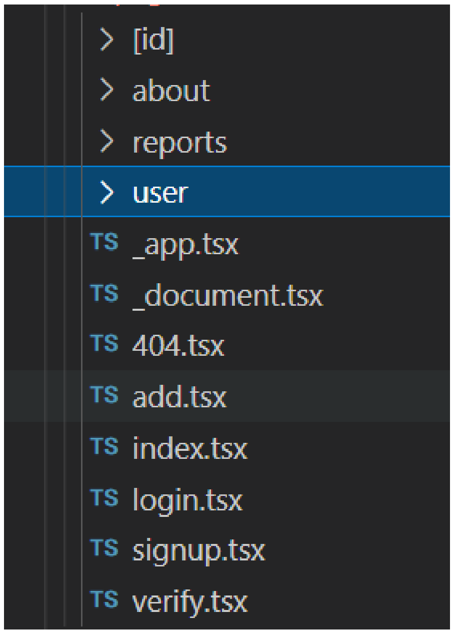
All the frontend code are saved under src folder



`Components` are tsx react elements that are imported into specific pages. This improves code readability and reusability

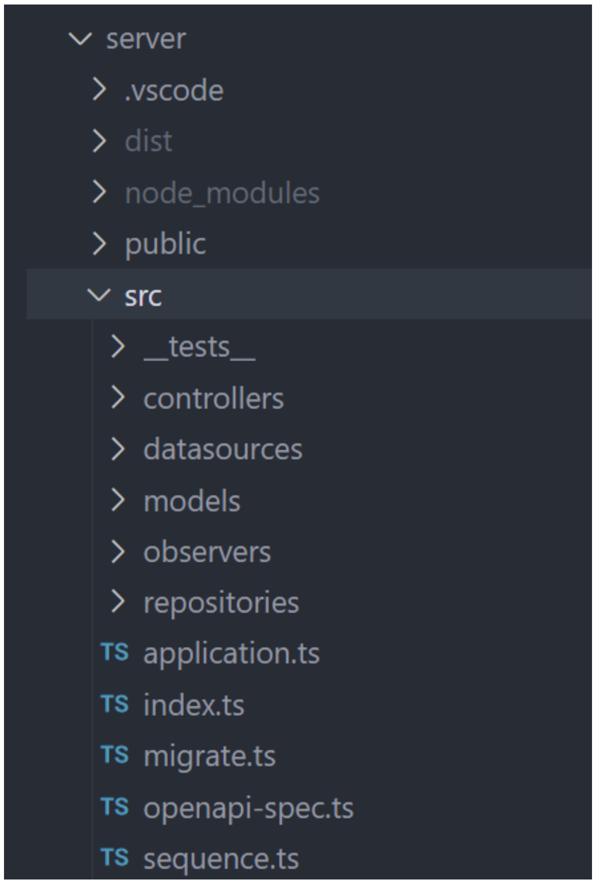
Global styles are stored under `styles` folder

Next.js renders tsx documents saved under pages folder automatically when that specific URL is asked for. The pages components are named specifically according to page.



Backend

The backend (Loopback 4) source code can be found in the /server directory. Loopback 4 classifies the source code into a number of classes which are very important concepts to grasp in order to better understand how Loopback 4 works. It is built on top of Model View Controller (MVC) architecture. So to start off, we have folders for models, controllers, and repositories. There could be many more folders depending on how complex a Loopback 4 application is. Our application has its code organized into the following folders:



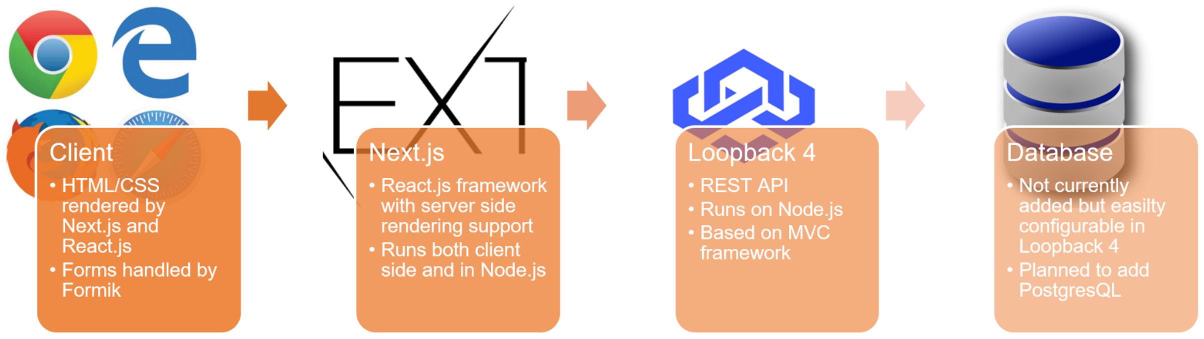
```
server
  .vscode
  dist
  node_modules
  public
  src
    __tests__
    controllers
    datasources
    models
    observers
    repositories
    TS application.ts
    TS index.ts
    TS migrate.ts
    TS openapi-spec.ts
    TS sequence.ts
```

Models are located in the “models” directory inside “server/src”. These describe the table structure of the data entities in our REST API. Each model is a TypeScript class in which properties are decorated with TypeScript decorators proved by Loopback 4. These decorators add additional functionality to these models when the application boots.

Because all of these files and folders simply follow the Loopback 4 concepts and conventions, we really recommend developers to read Loopback 4 documentation: https://loopback.io/doc/en/lb4/Inside-LoopBack-Application.html

Backend-frontend integration

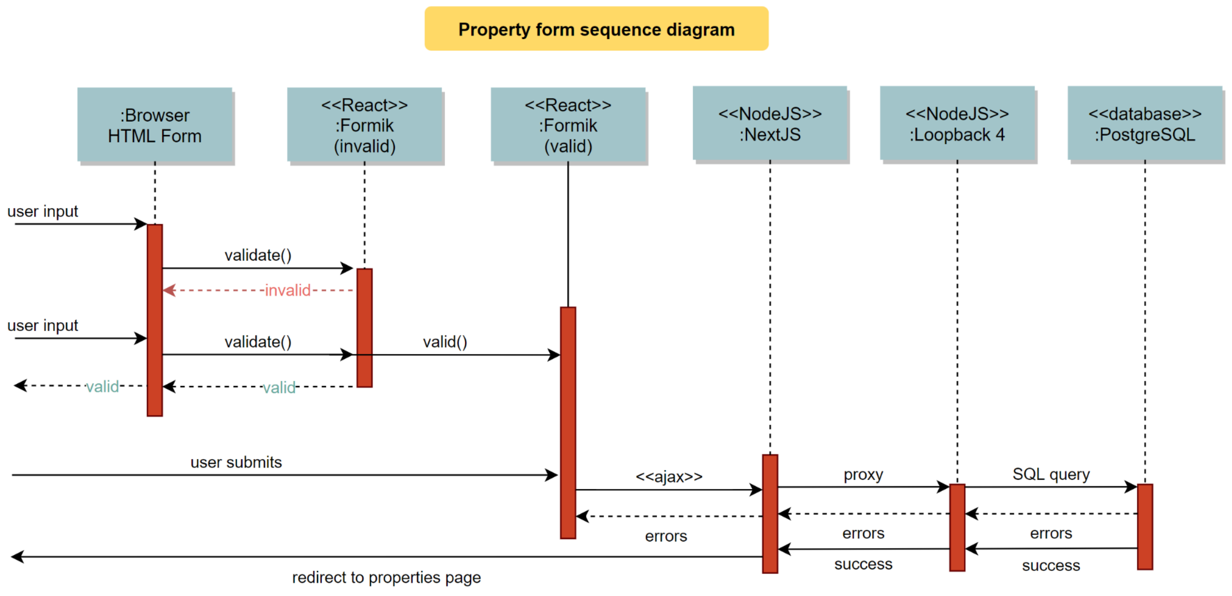
The following diagram summarizes our technology stack. The web application uses a multi-tier architecture i.e. the backend itself is divided into multiple layers i.e. Next.js, Loopback, and database. It is important to point out here that the application currently uses a volatile in-memory database and will be easily configured in Loopback to use PostgreSQL.


Client
• HTML/CSS rendered by Next.js and React.js
• Forms handled by Formik
Next.js
• React.js framework with server side rendering support
• Runs both client side and in Node.js
Loopback 4
• REST API
• Runs on Node.js
• Based on MVC framework
Database
• Not currently added but easilty configurable in Loopback 4
• Planned to add PostgresQL

The client-side HTML/CSS and Javascript are generated on the server-side by Next.js which runs on Node.js. Next.js itself is based on the React.js library which allows us to build easy interactivity into our components that build up the user interface.

Since it is a dynamic web application, our app depends on the REST API that is served from the Loopback 4 backend. However, all API requests from the frontend are routed through the Next.js application and not directly made to the Loopback application that is running on port 3001. This is due to CORS policy. The API routing is configured in /client/next.config.js

The following sequence diagram illustrates just one example of a typical action flow when the user interacts with the application. In this case saving or editing of a new property listing.



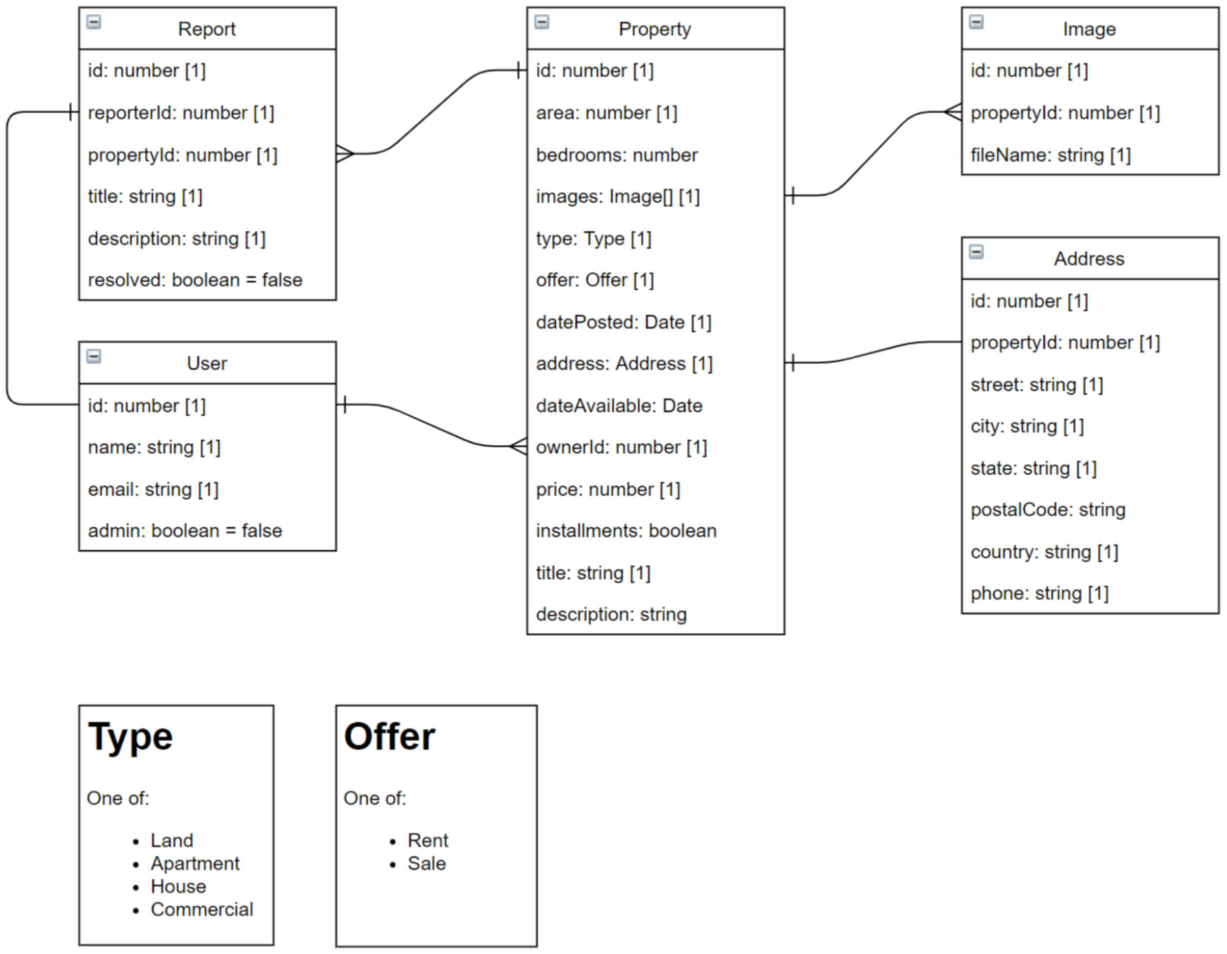
The following outline summarizes the steps illustrated in the diagram above:

- Whenever the user changes any input in the property listing form, Formik (which is a form handling tool based on React.js, which we use for form handling, and is discussed in detail below) listens to all change events in form input values.
- Upon change of any of the inputs, it calls the form validation method that we provide, and passes in the current form values. Our validation method checks the individual values against validation rules, in some cases, we have used Yup.
- In case the form is invalid, we provide Formik with an object of errors.
- Our customized form components (at /client/src/common/components/lib) that use Formik React hooks, detect if a validation error has occurred in the input in question and display an error message.
- Form submission is prevented if the form is invalid but when the form is valid, the form submission is allowed.
- When the user submits the form, a POST request is made to localhost:3000 (which is the Next.js application) with the form data in the request body.
- Next.js routes this request to the backend REST API which is served by Loopback 4.
- In our current implementation, the Loopback 4 application simply uses in-memory DB for easier testing, however, in production this will be a PostgreSQL database.
- As it can be seen from the above illustration, in various layers of this sequence, the process may fail due to errors and different stages. In case of an error that might be thrown at various stages, the action of saving the property may fail and will be investigated. However, with proper validation and testing, the process is most likely to succeed, in which case an HTTP 200 or HTTP 204 response will be returned, signifying success in saving the form.

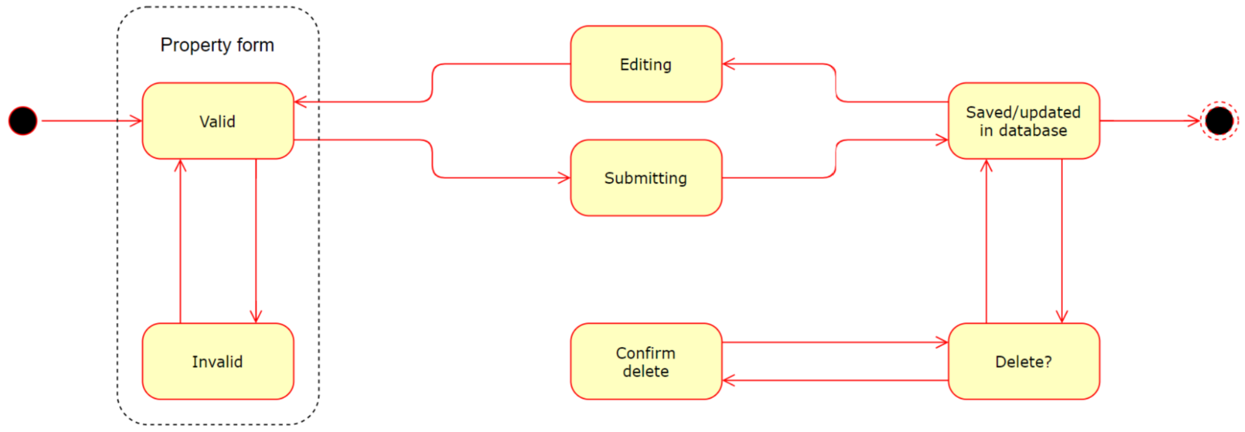
- The front end properly does this error handling and displays a success message only if the submission is successful. Please see the error handling here: /client/src/common/components/PropertyForm.tsx

Entity relationship diagram

The following entity relationship diagram illustrates the data model for the REST API. The app in its current implementation doesn't support addresses and image upload. However, the rest of the data model has been implemented.



The following state diagram summarizes the property life cycle:



CRUD user rights

The following decision table displays which types of users can create, read, update and delete (CRUD) which types of properties.

Property CRUD rights

User	Flag	Create	Read	Update	Delete
Logged out	None	None	All	None	None
Basic	All	All	All	Own	Own
Verified	All	All	All	Own	Own
Admin	All	All	All	All	All

User CRUD rights

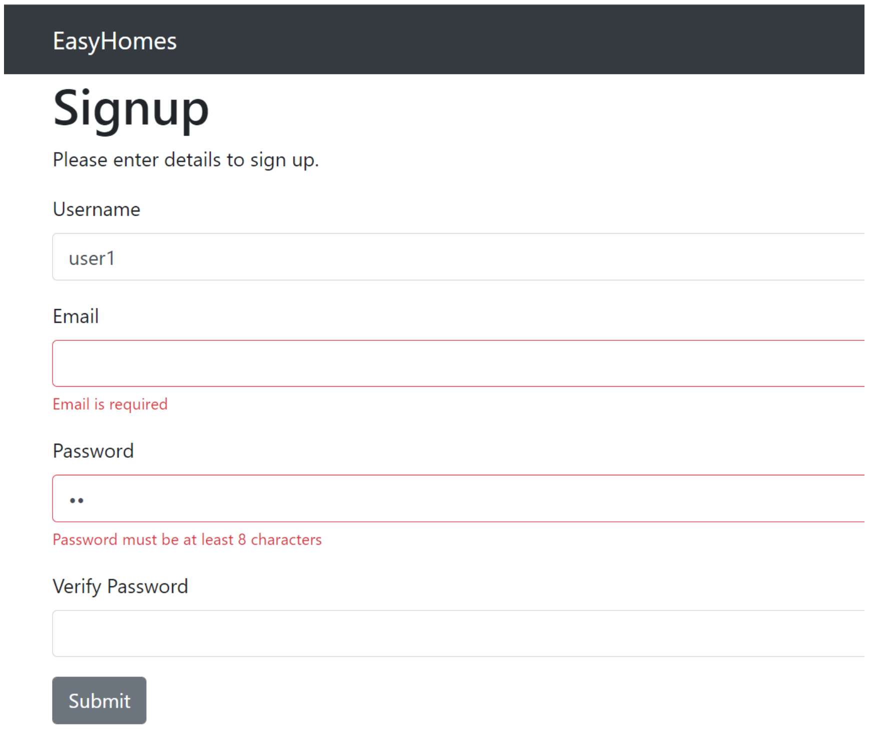
User	Basic	Verified	Admin
Logged out	Yes	No	No
Basic	Yes	No	No
Verified	Yes	No	No
Admin	Yes	Yes	No
Db Admin	Yes	Yes	Yes

Error handling

Both the backend and the frontend effectively handle errors using techniques such as try-catch blocks, type checking, custom fetch method on the frontend the throws errors if the response is not received correctly, using Loopback 4 error object, using Next.js Error component to display errors to the user in a friendly way, etc.

We validate the forms so that the user has timely feedback about validation errors. The form submission is prevented if it is invalid so that the user does not lose the data that they filled into the form.

These validation errors are updated in real-time as the user changes the form data.



EasyHomes

Signup

Please enter details to sign up.

Username

user1

Email

Email is required

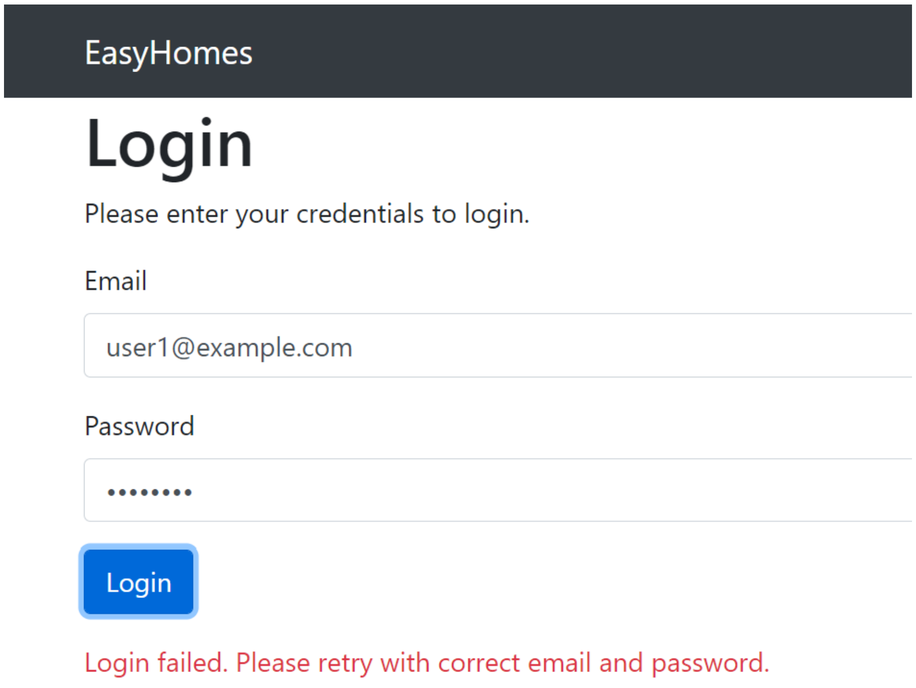
Password

••

Password must be at least 8 characters

Verify Password

Form submission error may happen even if the form is valid e.g. due to a network error. In such a case we catch the error using a try-catch pattern to display an error message and prevent navigation so that the user does not lose their form input data.


EasyHomes

Login

Please enter your credentials to login.

Email

user1@example.com

Password

••••••••

Login failed. Please retry with correct email and password.

Form handling

Forms are handled on the front end using Formik. It is a JavaScript library for React.js that comes with a lot of useful tools for handling React.js form, validating user input, handling form submission, etc. It provides useful React hooks such as useField() that we have used in our components to make them intelligently work with Formik Context.

For example, here is the code for our custom InputText component located at /client/src/common/components/lib/InputText.tsx.

```
import { useField } from 'formik';
import { FormFeedback, FormGroup, Input, Label, InputProps } from
'reactstrap';

interface Props extends InputProps {
  label: string;
  name: string;
}

export function InputText(props: Props) {
  const { label, name, type, ...otherProps } = props;
  const [{ value, ...field }, meta] = useField(name);

  const invalid = meta.touched && meta.error;
  return (
    <FormGroup>
      <Label>{label}</Label>
      <Input
        type={type ?? 'text'}
        invalid={!!invalid}
        value={value ?? ''}
        {...field}
        {...otherProps}
      />
      {invalid && <FormFeedback>{meta.error}</FormFeedback>}
    </FormGroup>
  );
}
```

We have used this useField Formik hook which provides us with Formik methods and other values that plugin into the component to make it work with Formik API. useField identifies the input using its “name” prop and automatically gets and sets the form values.

We also simplified how labels are displayed with each form input but encapsulating:

- the label,
- the input, and
- any related validation error message

in a single component. This significantly reduced the amount of markup that we later needed to write when creating the forms.

These Formik enabled form inputs are located in the /client/src/common/components/lib directory. When used in a form, we just have to wrap these in a FormikProvider component that has Formik context passed in as a value prop. Formik then automatically discovers these inputs and reads values from them as the user inputs data.

Formik then provides this data that we can then use to validate it against our validation rules or Yup schema and display any error messages. If the form is invalid, Formik will prevent form submission. This is so that the user does not lose the data that they have input into the form.

User session management

In this section, we briefly discuss how user sessions are managed in the front end of the application. Because Next.js is built on top of React.js, we make use of React Context API and use a custom hook. The relevant code is located at /client/src/common/utils/useSession.ts.

We would like to briefly discuss how it works. But first, let us introduce JSON Web token (JWT) is an encrypted string / token sent by backend to the client upon a successful login. The server doesn't need to keep any session history because it can rely on the encryption to ensure that next time the user makes a request with the token, the token may be verified using the private key that is stored on the server. Upon login, this JWT i.e. the token is saved on the client side using JavaScript localStorage API. Now that this has been explained, let's go back to useSession.ts:

```
import { useEffect, useState } from 'react';

export function useSession() {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  const loadSession = async () => {
    const jwt = localStorage.getItem('jwt');
    if (jwt) {
        // verify if the session is valid using a REST call
      } else {
        // destroy session
        localStorage.removeItem('jwt');
        setUser(null);
      }
    } else {
      setUser(null);
    }
    setLoading(false);
  };

  useEffect(() => {
    loadSession();
  }, []);

  const updateSession = () => {
    loadSession();
  };
  return [user, updateSession, loading];
}
```

What it basically does is, when the website first loads, it checks if the browser has a JWT token in JavaScript localStorage web API. If yes, it verifies this by doing a REST call to the backend. If the session is verified, a user object is set which has details such as username, email, realm i.e. whether an admin or a verified user or unverified user.

This code is a custom React.js hook and can be used in any component. However, because the user session is application-wide, we use this to set the React state at the application level and then make the session object available throughout the frontend application using React.js Context API. This is done in /client/src/pages/_app.tsx

Let's take a look at the following snippet:

```
// imports

export const SessionContext = createContext<any>(null);

function MyApp({ Component, pageProps }: AppProps) {
  const [user, updateSession, loading] = useSession();
  // more code

  return (
    <>
      {/* more code */}
      <SessionContext.Provider value={{ user, updateSession }}>
        <Layout>
          <Component {...pageProps} />
        </Layout>
      </SessionContext.Provider>
    </>
  );
}
export default MyApp;
```

First, we create a SessionContext object and export it, so that it is available to be imported into other components of the application. Next, we call the useSession hook to create a user state at app.ts level. The user object and updateSession function are then passed down as value prop to SessionContext.Provider. This makes these values available throughout the application using Context API.

Finally, we use this session context wherever we need it. For example, on the “New” property form page, we need the user to be logged in. So we grab the user object from the SessionContext and then allow the action, if the user is defined and disallow access if the user is not defined i.e. not logged in. The following snipped shows this:

```
// imports

const title = 'Add a new property';

export default function AddProperty() {
  const { user } = useContext(SessionContext);

  if (!user)
    return (
      <Error statusCode={401} title="Please login to post a property" />
    );

  return (
    <>
      <h1>Add property</h1>
      <PropertyForm />
    </>
  );
}

AddProperty.title = title;
```

There are many more slightly complex use cases of useSession usage. We hope with the above example, you will be able to understand those complex use cases as well.

Running the project locally

The application is hosted at https://cm2020-agile-software-projects.vercel.app. And the backend at https://boiling-springs-75569.herokuapp.com/. However, it may also be run on most Windows or Linux-based operating systems that have Node.js installed.

Pre-requisites

The following must be installed before any development work may be done:

1. Node.js version 14 from https://nodejs.org/en/. Node.js may already be installed on your PC, to check, run command ‘node -v’ in your terminal, and ensure that the version you have installed is 14 or later. To easily manage the node version on your system, we recommend managing the Node.js version using Node Version Manager (NVM).
2. Node Package Manager (NPM). This is usually installed with Node.js.
3. A terminal, with the ability to run two instances: one for backend and one for frontend.
4. A clone of the source code: irfanullahjan/cm2020-agile-software-projects (github.com). Source code may be obtained either in zip format from the above link. Alternatively, the repository may be cloned using Git (which must be installed). To clone the repository, please run the following in your terminal:
   ```
   git clone git@github.com:irfanullahjan/cm2020-agile-software-projects.git
   ```

Optionally, you may also want to install Docker if you want to run the application containerized. VSCode is recommended for development.

Running the application

The application is made up of two major components, one for backend API and the other for front-end functionality:

The backend REST API is provided by a Loopback 4 based application. The front-end of the website depends on it, which means that this Loopback application must be run before trying to run the frontend part.

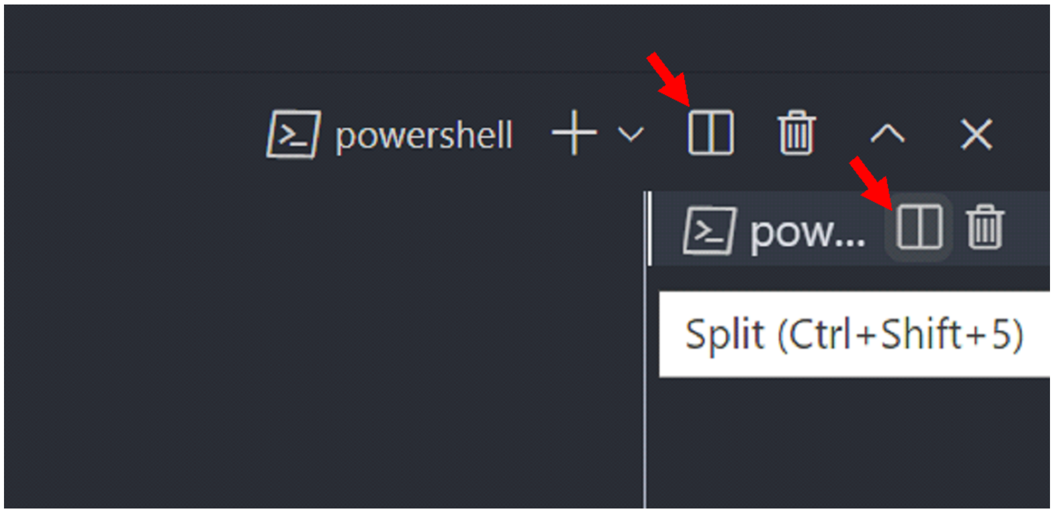
It is located in the /server directory.

This is the frontend of the website i.e. the one containing the UI and is mostly client-side, though there is server-side rendering as well which is supported by Next.js.

It is located in the /client directory.

To run the application locally, please clone the repository to your machine and then run the following commands in the given order (in the directory where you cloned the repo).

Because the application is composed of two parts i.e. the backend (Loopback) and the frontend (Next.js), we need to run both to be able to use the application. Both of these need to be running in separate terminals. If you are using VSCode please try its split terminal feature to view both frontend and backend logs simultaneously.



Install and run the server:

```
cd server
npm install
npm run start # or simply `npm start`
```

These commands will open the /server folder, install the dependencies required for the application’s backend i.e. Loopback 4 based, and start the backend API.

Install and run the client (in a separate terminal):

```
cd client
npm install
npm run dev
```

These commands will open the /client folder, install the dependencies required for the application’s frontend i.e. Next.js based. It is important to note that these two must be running simultaneously in separate terminals.

You may also do `npm run build` and then `npm run start` on /client directory to start optimized production build on the frontend.

Accessing the application

The application should now be running at http://localhost:3000

To access the backend i.e. REST API explorer, go to http://localhost:3001/explorer

Project management methodology

The team adopted agile methodology, specifically, weekly scrum sprints as the style of development for the following reasons:

1. Core functionalities were deemed more important than the technology itself. Fundamentally the application should solve the issues (reducing real estate fraud in Kenya) that it is focused on.

2. Since the development of the prototype and minimum viable product is centred around users, the team needs to be flexible to try new designs and patterns that have a good ‘product-market fit’ rather than creating comprehensive documentation.

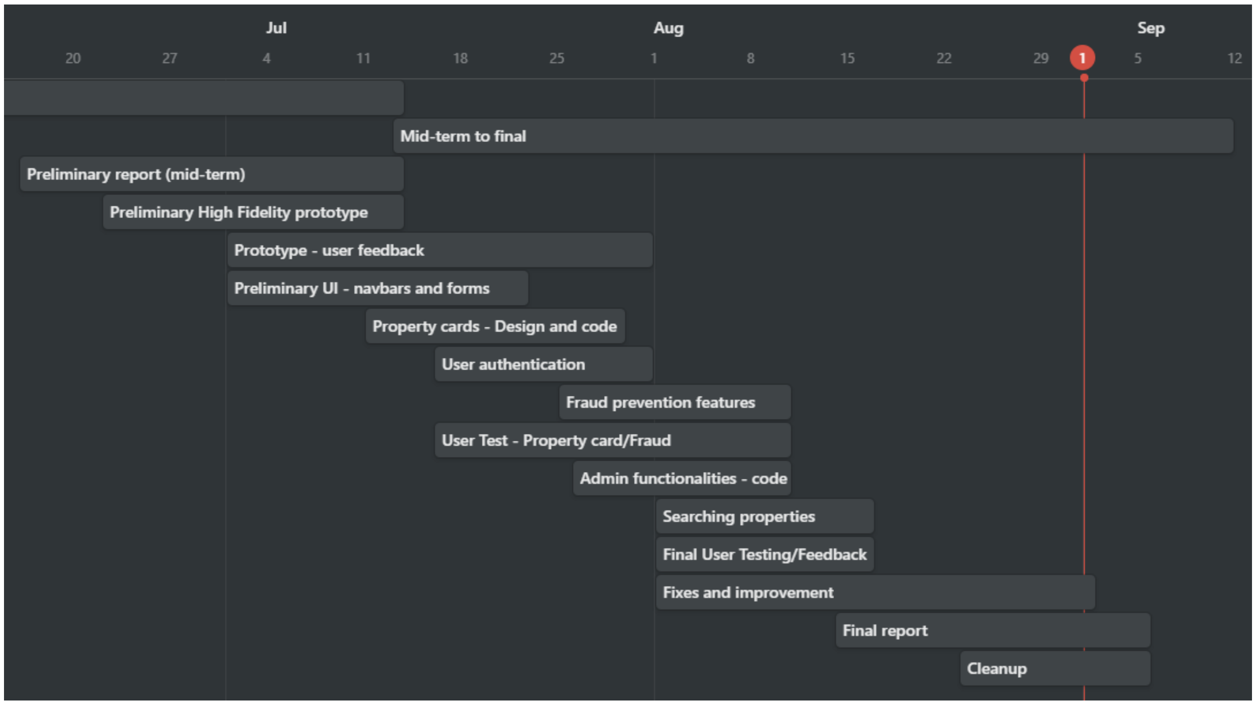
3. The users are consulted every month, after a new functionality or design is added, to collect feedback.

4. Methods such as A/B testing and survey forms were used to collect feedback on functionalities and designs that provided a positive user experience. The designs were then discussed on the weekly meetings and iterated in weekly sprints

The team’s style of development aligned with the Agile manifesto and user-centred design methodologies which in our case complemented each other.

The team created a Gantt chart that charted out the core functionalities of the application mentioned in the scope. This gave a good visual representation of the project schedule and things to focus on during each weekly sprint.

The Gantt chart was created in Notion.so which is an online collaborative workspace software where the team stores the notes, developer wikis, charts, and other resources. The tasks timeline started immediately after the midterm submission and ended before final submissions. Tasks include various designs, application functionalities, and user testing phases.



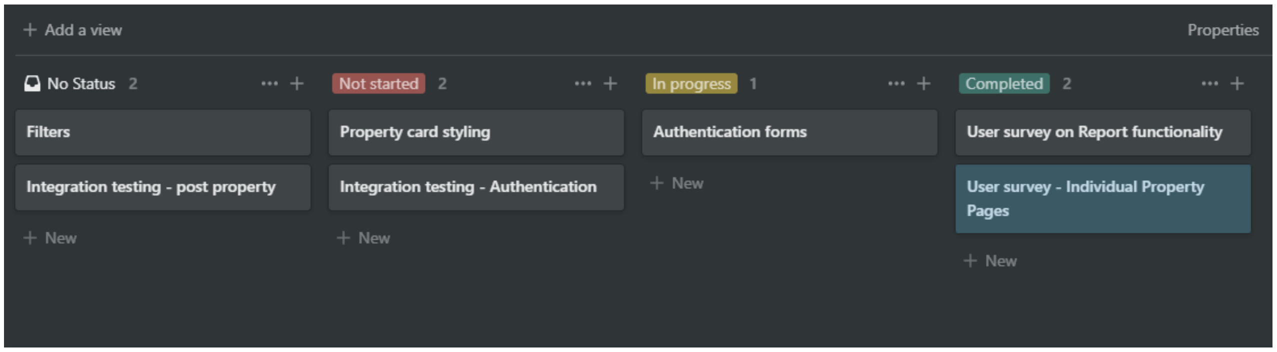
The team adopted a unique style to implement Scrum methodology where weekly sprints were conducted with weekly tasks being split and tracked using a kanban board functionality(inside Notion.so). This team adapted this style to have a better understanding of the product backlog and achieve the features necessary for the minimum viable product in small increments. This also provided adaptability to change according to the user reviews.

Each task was handed off to each team member with respect to their area of expertise. The code development work was collaborated using a central repository (GitHub). For code development, each feature was developed by a team member in a feature branch and later merged into the main branch after code review.

One of the team members acted as a scrum master. Sprints were organised on a weekly iterative basis because the team members found it efficient with consideration to their schedules. Team members decided the scope of each sprint during the weekly video meet-up. Unfinished work (sprint backlog) was assigned to the consecutive sprint if the team members deemed it as important. Although daily scrum standup meetings were not conducted, the team kept each other in the loop by updating their tasks in Kanban boards and the Slack channel.

Weekly meetings also helped with scrum review, where the team evaluated the features we managed to implement if a new design was needed based on the User testing and task assignments. The application was demoed during these reviews.

A sprint retrospective meeting was conducted during the last week to review how effective this methodology was, plans for further development of the project, and personal opinions.



No Status	Not started	In progress	Completed
Filters	Property card styling	Authentication forms	User survey on Report functionality
Integration testing - post property	Integration testing - Authentication		User survey - Individual Property Pages

Task tracking using kanban board functionality inside Notion.so

Team composition and roles

The team is made up of 4 members:

1. Abigail Gomes Leitao
2. Adele Gikonyo
3. Irfanullah Jan
4. Ivan George

Throughout the project, each of us took our own loosely defined role. These roles developed slowly as the team discovered the strengths and weaknesses of each of the team members:

1. Abigail contributed primarily to the design and competitor research:
    a. Building and amending initial prototypes for the final application
    b. Researching the biggest competitors and gathering information about similarities and differences between them
    c. Contributing towards the evaluation of designs throughout production.
2. Adele's role was mainly that of business analyst and she contributed with:
    a. Market research, conducting and evaluating user surveys, and incorporation of user feedback into the design process
    b. Building and designing the initial prototypes for the design of the application
    c. Gathering and tracking requirements from potential users of the application
3. Irfan mostly overlooked the development of the application and contributed with:
    a. Programming the backend API using Loopback 4
    b. Contributing in the frontend design and development using Next.js
    c. Writing tests for the backend and frontend

    d. Deploying the application to the test environment i.e. Vercel and Heroku

4. Ivan performed the role of scrum master and contributed with:
    a. Development and design decisions
    b. Styling and development of property cards and home page
    c. Usability and functionality research
    d. Tracking tasks and reviewing sprints

Essentially this whole project only became possible due to an excellent team effort. We communicated regularly and provided feedback to each team member about their progress. Evidence of this is our Slack channel that remained very active throughout this period, with tons of updates about the progress of the project and the feedback from the team members.

In our regular weekly meetings, we demonstrated the application and shared useful insights about how to better solve the difficulties any of the team members faced with the project that was slowing down their progress.

Weekly Meeting Minutes and Sprint Reviews

Note: Detailed minutes of meetings have been attached separately.

Week 5 - 16/5/2021
Decided on the project, its relevance, and the basic features needed. Using technology to make the process of selling or finding properties in Kenya more transparent and easier.

Discussed various frameworks and tradeoffs, like team expertise in that framework. Decided on Next.js for the frontend and Loopback for the backend.

Week 6 - 21/5/2021
Discussions of UML diagrams and user flow of the application. Deciding on features needed for low fidelity diagrams. Demoing GitHub as the central repository. Questionnaires on market research. Researching competitions and discussions

Week 7 - 30/5/2021
Discussions of results from questionnaires and insights derived. Fixing the scope of the project.

Week 8 - 07/6/2021
Created the app and deployed it to Github. Discussions on various UI elements and how to iterate with user feedback. Looked into various authorisation services such as passport.js. Decided not to use Docker to containerize the application.

Week 9 - 13/6/2021
Finalised UI mockups. Discussions on responsive design
Finalised on buttons, filters, element layouts

Week 11 - 27/6/2021
Finalisation of scope
Different layouts and pages were discussed
CRUD access for different users and how to develop it inside the app

Week 13 - 11/7/21
User testing frameworks and methods
Discussions on which heuristics methods to follow
Elements that should be A/B tested
Creating forms for users to submit their response

Week 14 - 19/7/21
Discussions on how to structure frontend code based on Next.js best practices
How to structure various components and what all pages to create
Discussions on Next.js

Week 15 - 25/7/2021
Creating tasks for each member to take over like: view-property page, property-cards, about-page
Discussions on how to implement authentication for users

Week 16 - 1/8/2021
UI discussions on admin privileges and pages
Remove button functionality and UX implementation

Week 17 - 8/8/2021
Created tests for navigation with cypress.js
Frontend tests for authorisation to only allow user status based activities
Added some typescript code

Week 18 - 15/8/21
Discussions of user flow and functionality based on user reviews

Week 19 - 22/8/2021
Final iterations on the product
Removing verified tags from property cards and moving them to users
Creating hamburger menu filter for mobile pages

Week 20 - 26/8/2021
Discussions of challenges faced during production.
Scope of development, and how the project aligns with the initial scope set by the team.
Plans for future development.

Collaboration using GitHub

Every team member cloned from the master repository and created a local repository of the project. A local repository was created by pulling from the cloned or main repository. All changes were made after creating a “feature-branch” in each member's respective local repository. Then these changes were committed and pushed to the respective member’s forked repository

Then create a pull request to the main remote repository. The code is then reviewed and merged by the maintainer of the repository. Once the feature is included, the feature branches are deleted.

Git logs have been attached with this report.

Automated testing

Because both the frontend and the backend of the application are developed using TypeScript, it serves as the first line of defense against errors because it captures potential errors at compile time. In a sense, TypeScript replaces basic unit testing.

Because of the complexity and amount of work required to get the minimum viable product ready, we couldn't afford to allocate too much time to do a lot of testing. We still were able to add a number of tests to both the backend and the frontend.

Backend

Loopback 4 testing is based on their own implementation of the popular JavaScript testing framework Mocha which adds support for unit tests. We used this built-in support for testing and added some acceptance tests which is a form of black-box testing, which means that we are not concerned with the internal components of the system as long as the inputs and the outputs of the system are consistent with our requirements.

So, in black box testing, we test the system with a set of inputs and we consider the tests as passed if the corresponding set of outputs matches the ones already known to us based on our understanding of the requirements.

The backend tests can be found in the /server/src/__tests__/acceptance directory. To run those tests, please navigate to /server directory and then run:

```
npm run test
```

We added tests, for example, to ensure that the endpoints are working and that the test user accounts and admin account is available on startup.

Frontend

The frontend which is Next.js may either be tested using Cypress, Jest, or React Testing Library. We chose Cypress because it is a popular end-to-end integration testing framework. It is also well supported and documented by the Next.js team.

We wrote integration tests for navigation, for example. These tests ensure that different pages of the frontend application are accessible. This is performed by Cypress in an automated way using a web browser engine.

Integration tests, test the system as a whole single application. This means that when the tests are run, both the backend and the frontend of the application must be running and properly integrated.

We also added frontend tests for authorization. These tests ensure that users are only able to do what they are allowed to do. For example, unauthenticated users are not able to post properties or report properties. Non-admin users are not authorized to perform actions that only admins are allowed to perform. These and other frontend tests are located at /client/cypress/integration directory.

To run these tests, the developer should navigate to /client directory and then run the following command:

```
npm run cypress
```

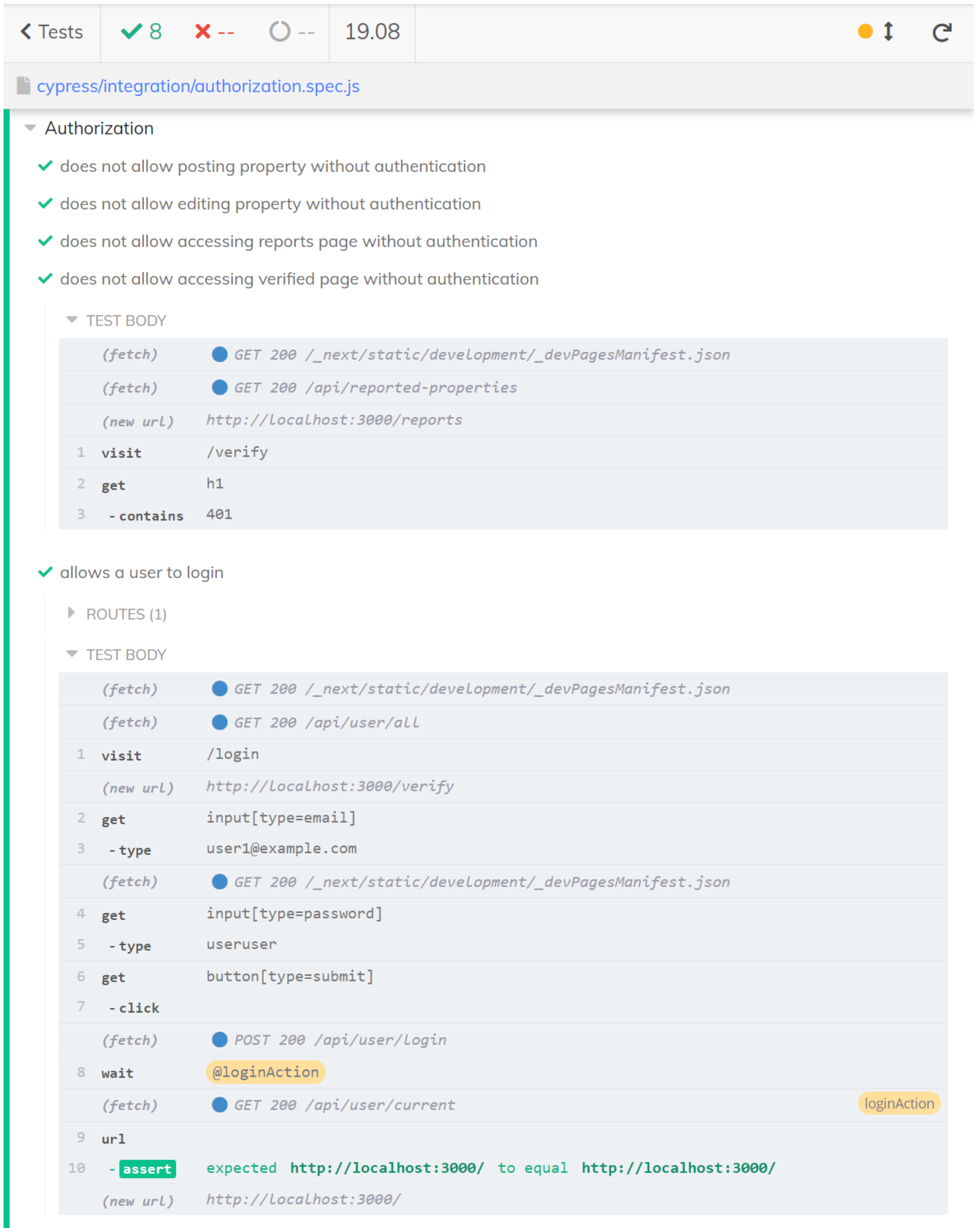
The following is just an example frontend integration test:

```
describe('Authorization', () => {
  // More code ...
  // Excerpt

  it('allows admin user to access reports page', () => {
    cy.intercept('/api/user/current').as('loginAction');
    cy.visit('/login');
    cy.get('input[type=email]').type('admin1@example.com');
    cy.get('input[type=password]').type('Admin!234');
    cy.get('button[type=submit]').click();
    cy.wait('@loginAction');
    cy.visit('/verify');
    cy.get('h1').should('not.contain', '401');
  });
});
```

The following is an example backend acceptance test:

```
describe('UserController', () => {
  // More code ...
  // Excerpt

  it('admin user available on start up', async () => {
    const userRepository = await app.getRepository(UserRepository);
    const admin = await userRepository.findOne({
      where: {email: 'admin1@example.com'},
    });
    expect(admin).to.containEql({email: 'admin1@example.com'});
  });
});
```

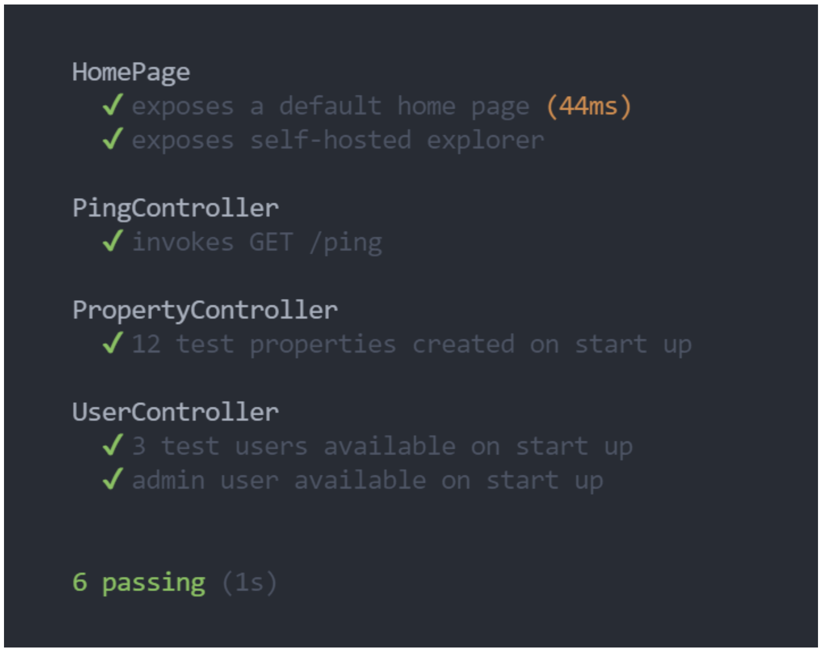
The following screenshot shows some of the frontend tests that we ran:



```
< Tests   ✓ 8   ✗ --   ○ --   19.08

cypress/integration/authorization.spec.js

▾ Authorization
  ✓ does not allow posting property without authentication
  ✓ does not allow editing property without authentication
  ✓ does not allow accessing reports page without authentication
  ✓ does not allow accessing verified page without authentication
    ▾ TEST BODY
        (fetch)      ● GET 200 /_next/static/development/_devPagesManifest.json
        (fetch)      ● GET 200 /api/reported-properties
        (new url)    http://localhost:3000/reports
        visit        /verify
        get          h1
         - contains  401

  ✓ allows a user to login
    ▸ ROUTES (1)
    ▾ TEST BODY
        (fetch)      ● GET 200 /_next/static/development/_devPagesManifest.json
        (fetch)      ● GET 200 /api/user/all
        visit        /login
        (new url)    http://localhost:3000/verify
        get          input[type=email]
         - type      user1@example.com
        (fetch)      ● GET 200 /_next/static/development/_devPagesManifest.json
        get          input[type=password]
         - type      useruser
        get          button[type=submit]
         - click
        (fetch)      ● POST 200 /api/user/login
        wait         @loginAction
        (fetch)      ● GET 200 /api/user/current                          loginAction
        url
         - assert    expected http://localhost:3000/ to equal http://localhost:3000/
        (new url)    http://localhost:3000/
```

Note: There is an issue with Cypress that causes some tests to fail randomly, even though there is nothing wrong with the UI. Only that the Cypress triggers click before the UI is ready. This is a known issue and is under consideration to be fixed. Please see this Github issue: https://github.com/cypress-io/cypress/issues/7306

The following screenshot shows the results of backend tests that we ran:



```
HomePage
  ✓ exposes a default home page (44ms)
  ✓ exposes self-hosted explorer

PingController
  ✓ invokes GET /ping

PropertyController
  ✓ 12 test properties created on start up

UserController
  ✓ 3 test users available on start up
  ✓ admin user available on start up


6 passing (1s)
```

User testing

“Usability is the extent to which a system, product, or service, can be used by specified users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use.”

As a team, we all had our own house-hunting experiences to draw from. We started by setting aside any assumptions we had on what users might need from an app. We decided to let the users inform the problem statement first and we conducted different surveys and tests during different phases of the development of the EasyHomes application.

In order to stand out from competing applications, we wanted to build an application that not only had all the features that the users wanted but also an application that had a strong level of usability. Usability refers to not just the aesthetics of the user interface (Ferre et al. 22-29). Usability refers to how users relate to the software in question with the five basic attributes of usability being: learnability, efficiency, user retention over time, error rate, and satisfaction.

While building our application we tried to follow the Nielsen (2020) principles as closely as possible and towards the end of the development process, we conducted a usability survey to measure how well we did.

Below is a list of the Nielsen (2020) principles that guided our development of the application:

1. Visibility of system status
2. Match between system and the real world?
3. User control and freedom
4. Consistency and standards - similar styling of buttons, not overly busy
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognise, diagnose and recover from errors
10. Help and documentation

As much as possible we tried to be consistent in our styling and went for a minimalist design. We strove to be consistent in our entire approach so as to help our users become more familiar with our app as they use it without having to learn new representations of the same actions. We, therefore, used familiar user-flows, icons, colours and menu flows. We recognise that human attention is limited and therefore choose recognition over recall so as to reduce short-term memory load.

Simple error handling - We tried as much as possible to design the system to be fool-proof in terms of users encountering errors but we recognise that unintentional errors can happen and we wanted to ensure that we provide our users with simple and intuitive instructions on how they can solve their problem as quickly and pain-free as possible. For example in our forms, if the user's form is not accepted due to missing information then we flag the sections with the missing information

The only one of Nielsen's (2020) principles that we did not want to adopt at this stage was the match between system and real-world and making the application resemble a real-life agent experience. We decided that this was not really necessary and that at a later stage of the application we could add more stylistic elements to make it match a real-life agent experience without compromising on the minimalistic aesthetic.

Towards the end of the development process, we conducted a usability survey and our application scored a total of 70. We sampled 6 users and we gave them the System Usability Scale (SUS) survey which provides a "quick and dirty" reliable tool for measuring usability. The questionnaire consists of 10 questions each with 5 options for users to select ranging from strongly disagree to strongly agree. We decided to test only 6 users because we felt that this was the optimal number to catch any errors in the application without wasting people's time.

We tested the application with real users - we got real users to use the software and evaluate through various metrics how the system scored from a usability perspective.
provide feedback for further iteration of the application.
We created a survey to test
We conducted system useability surveys

Iterative design

Throughout the design, development, and testing phase, we continuously kept on improving the design based on the feedback we received from the users.

We mostly adopted agile processes and ways of working which helped to increase productivity and innovation within the team. This meant that there was constant collaboration within the team.

User feedback was in multiple forms, including online surveys which were taken by users at multiple stages of the application development:

1. Design phase
2. Development phase
3. Testing phase

Design phase

The first step in the design phase was to come with some initial mockups. The team decided to brainstorm and each of us came up with their own mockups for various pages of the application.

We revised our designs based on initial user feedback to ensure that the application will have the features and functionality that the users need and that the usability will best match the use case of the users. This ensured that later in the development phase, we will have minimal changes to make. The aim was to get the designs right from the start.

Development phase

As the application developed, we shared screenshots with users to obtain their feedback on whether the application is progressing how they wanted it to be. Because the users didn’t have the tools and the knowledge to run and test the application in their own machines, we decided to demonstrate it using screen sharing using zoom. This meant they were able to get quite an authentic experience of how the application would work. We received a lot of useful feedback at this stage. For example:

1. When the application is loading content, we should use spinners so that the user has an idea that something is loading. This was added to pages as well as form submit buttons.
2. Forms such as signup and property posting/edit form should have validation with messages on each input, informing the user why the form is invalid. Users should stay on the page when the form is invalid so that they don’t lose their data that is yet unsaved.
3. We initially had properties shown at /properties page, however, the users requested to show the properties and filtering right on the home page, because

this is the main purpose of this application. They disliked having to navigate to the website and then clicking a link to go to the properties page every time.

Testing phase

Most of the work during final testing involved bug fixes and some improvements to the user experience that were not evident from the start. To facilitate easier testing by the users, we deployed the application to Vercel and Heroku, the former for the frontend and the latter for the backend. This application is currently running at:

https://cm2020-agile-software-projects.vercel.app

Please note that the initial page load may take time if no one has recently accessed the application because the Heroku dyno goes down when not used and may take a few seconds to restart. It should be faster after the initial load.

This enabled the users to easily test the application on their own device and then share the feedback using direct messages, as well as an online survey. At this stage, the users reported a number of bugs that we had not encountered before. The users shared feedback regarding how to improve user experience and UI. All this enabled us to make countless bug fixes and improvements near the end of the project.

Some of the notable fixes and improvements we made at this stage include:

1. Improvement to the “View Property” page. We improved, for example, the naming of the items and the date format. This was after we received valuable feedback from one respondent who commented:

   “When you click into "View", some field names and values aren't user-friendly. For example, createStamp could be changed to "added on" and "updateStamp" could be changed to "last updated". The actual date and time value appears as "2021-09-01T11:53:56.303Z" and could be changed to "September 1 2021 11:53" and the price doesn't have a currency value, just 700000 (or whatever the value is).”

2. We removed “verified” badges from properties cards because this was the most disliked feature in one of our final surveys. Because actually the users are verified and not the properties, and this confused the users. So we removed the badge from the properties cards and added the verified badge to the user name on the “View Property” page. (Please see Appendix 1) One user says:

   “*Verified properties don't really give assurance that it's verified. Need ratings based on different aspects like safety and so on which will give more credibility to the verified properties.There should be a feature which shows how many people are interested in a particular property while viewing the profile”*

3. We added a button that when pressed, clears all the filters applied to the properties search on the main page. This was after we received feedback from one of the user in our survey near the end of the project, which said:

   "Reset all filters button Property card not clickable. I want a click on the card to take me to the property page. Min max range can go to negative values. Also there should be ranges for quick selection. Can't figure out how to report property"

   This user was not able to figure out how to report a property, and this is something we need to improve.

4. Multiple users pointed out that the application doesn't show proper page titles on the browser. So, we added HTML page titles to all pages of the application. This is helpful because the users are able to see what is on the page e.g. if they are on another tab in the browser.

5. Showing spinners on form submission buttons to indicate to the users that the form submission is in progress. This is so that in case of slow network speed, the users don't keep pressing the submit button again and again. Therefore, it improves the user experience.

6. Users complained about the inability to delete their property listings. We added a way for logged-in users to delete their own properties either from the "My properties" page or the "View property" page.

Critical evaluation and discussion

Challenges

We believe our aim was quite high and developing such a feature-rich application is quite challenging. In fact, when we initially shared our proposed idea with one of the tutors, they commented:

“Is your project a software development that would build a solution similar to rightmove.co.uk from scratch? If yes, I would advise you to find out (and validate) what would be the MVP (minimum viable product) to be offered in Kenya (including the component to minimize frauds) as it would be impossible to develop a complete solution as Rightmove currently offers. Remember...they have been in the market "for ages" and they have an end-to-end marketplace covering many countries.”

The team did manage to complete the scope of the Minimum Viable Product but took more effort and time than estimated.

Another challenge the team faced was not all members of the team being experienced in the frameworks used for development. This was remedied to an extent by conducting weekly tutorials and explanatory sessions. Also assisting each other through Slack and GitHub.

Not being able to meet the users and stakeholders face to face due to gathering restrictions was a challenge, but video calling applications were used to mitigate this issue.

Our performance

We believe we performed very well considering the complexity of the product. We were able to mostly meet and in some cases exceed our initial goals regarding the minimum viable product. For example, we had three pages in our initial design, however, we created many more dynamic pages to facilitate easy navigation and browning of the content.

The users are able to sign up, log in, log out, post new properties, view their own properties, edit their own properties, search and filter all properties, report properties other than their own, browse properties by the user, see if the user is verified or admin.

Admins are able to (in addition to what a normal user can do above) view reported properties, take action (delete property) if needed, set users as either verified or unverified.

We were also able to iterate over the design and improve the usability and functionality of the app

Major flaws

In this section, we note down the major flaws in the application that must be fixed before the application is actually deployed for use by real users.

1. **Data persistence**

   To allow easier testing and development, the application currently uses an in-memory database instead of persisted on storage databases such as MySQL or PostgreSQL.

   Currently, we didn't connect a PostgreSQL DB so that the grader is able to test the application without having to run a database instance.

   However, switching to a persisted DB is straightforward and just requires configuring it in /server/src/datasources/db.datasource.ts

2. **API protection**

   Currently the backend API i.e. the Loopback 4 application is exposed. That is, the endpoints are not properly protected using authorization rules/matrix. This means that any user that is logged in, can directly call the API endpoints to manipulate data belonging to other users, they can delete data, update data and post data. This means that a normal user once signed up can use their authentication token to perform e.g. actions that only an admin should be able to perform. To fix this, we need to apply authorization checks to endpoints to verify the user's identity and the role and only allow actions only if they have the right to do so.

3. **Limited features**

   The application still likely doesn't meet the minimum needs that the customers in real estate need. For example, a number of customers pointed out the inability to contact the property agent, inability to view images and floor plans, etc. We have to admit, the application turned out to be too ambitious for a team for 4 in the time available to use.

Future development

Despite all the work that we did during the past few months, the web application still has quite a bit of room for improvement. Some of the things that should be at the top priority in future development include:

1. Ability to upload images and floor plans. This was one of the features a number of users prioritized at the top of our user survey. One user remarked.
   "*I think images and floor plans are essential features, not sure I would use the app until those were implemented!"*

2. Ability to contact the real estate agent. This could be simply an email address or phone number of the agent posting the property, or it could be a sophisticated built-in feature where each user would have an inbox within the application where they will be able to receive messages. In any case, it is one of the most important features. One of our user survey feedback says:
   "*The application so far is a good base app to build out more features like contacting estate agents or sellers, viewing last bid, last sold price, etc."*

3. Ability to book viewings directly on the application itself through a book viewing button. This will send a message to the server which sends a request to the user who posted the corresponding property listing.

4. Implementing an internal map and location using google maps API, so the users can get access to the location of the property or search for properties by clicking on the location.

5. Translating the user interface and implementing the translations using frameworks such as i18next. This is essential to make the application available to a wider audience.

6. Since the team members are not familiar with cloud technologies, gaining expertise in Amazon Web Services and hosting the site along with the database.

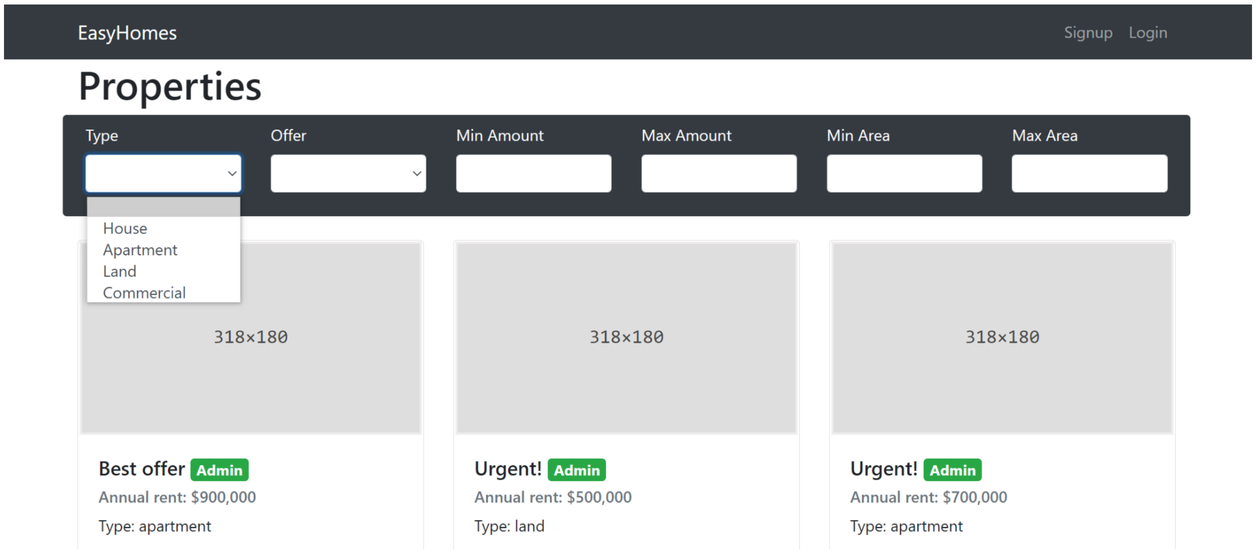
7. Ability to log in using auth0 authentication services which add sign up using social media accounts (Google, Facebook etc). This would make the user experience seamless and simple.

8. Licensing the app and open-sourcing the data so the Kenyan developer community can make suggestions and add features for further development.

Appendix 1: Iterative design screenshots

Properties search filter

Before



EasyHomes

Signup Login

Properties

Type	Offer	Min Amount	Max Amount	Min Area	Max Area

House
Apartment
Land
Commercial

318×180

Best offer Admin
Annual rent: $900,000
Type: apartment

318×180

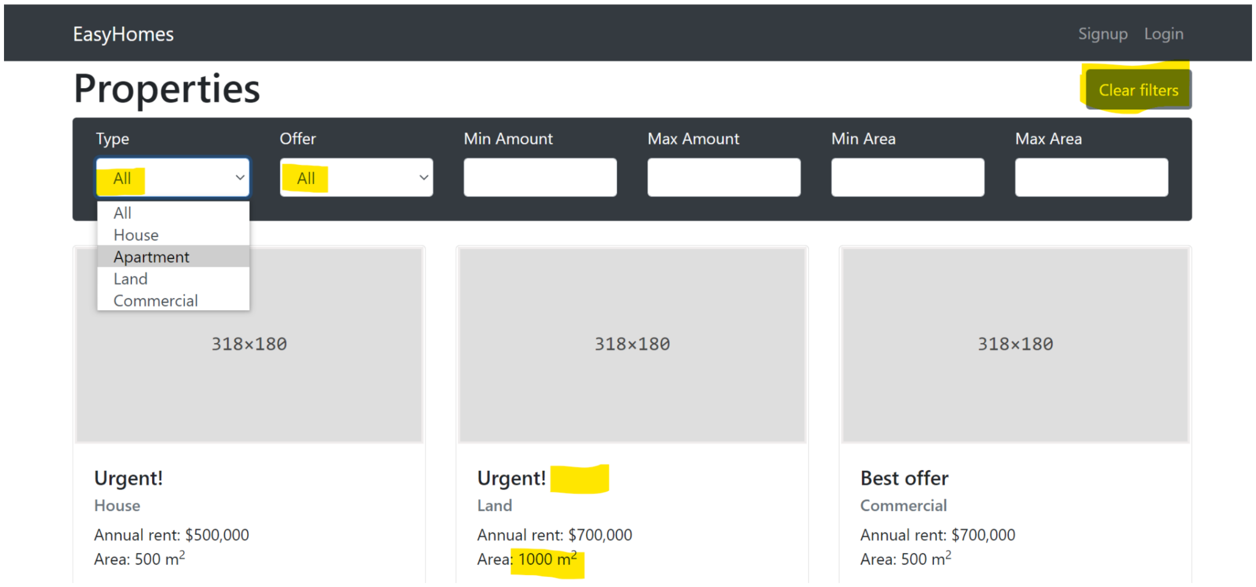
Urgent! Admin
Annual rent: $500,000
Type: land

318×180

Urgent! Admin
Annual rent: $700,000
Type: apartment

After

The verified badge is now shown on the View Property page.



EasyHomes

Signup Login

Properties

Clear filters

Type	Offer	Min Amount	Max Amount	Min Area	Max Area
All	All				

All
House
Apartment
Land
Commercial

318×180

Urgent!
House
Annual rent: $500,000
Area: 500 m^2

318×180

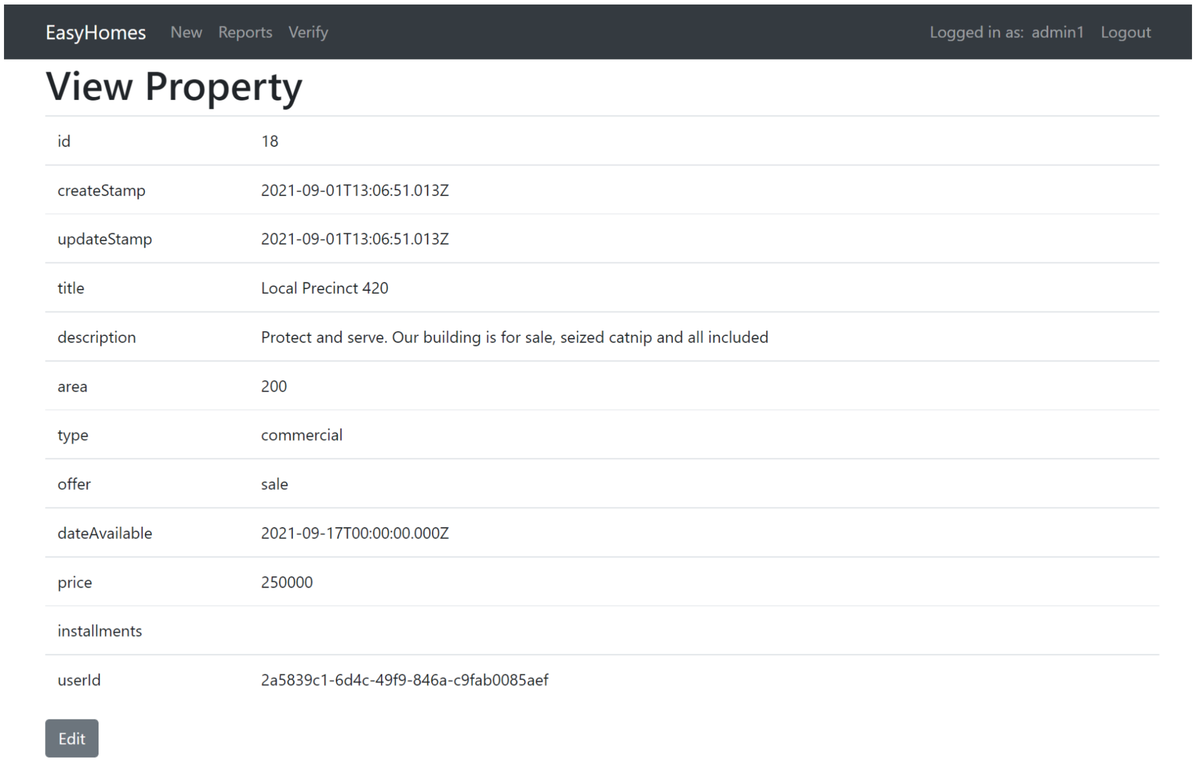
Urgent!
Land
Annual rent: $700,000
Area: 1000 m^2

318×180

Best offer
Commercial
Annual rent: $700,000
Area: 500 m^2

View property page

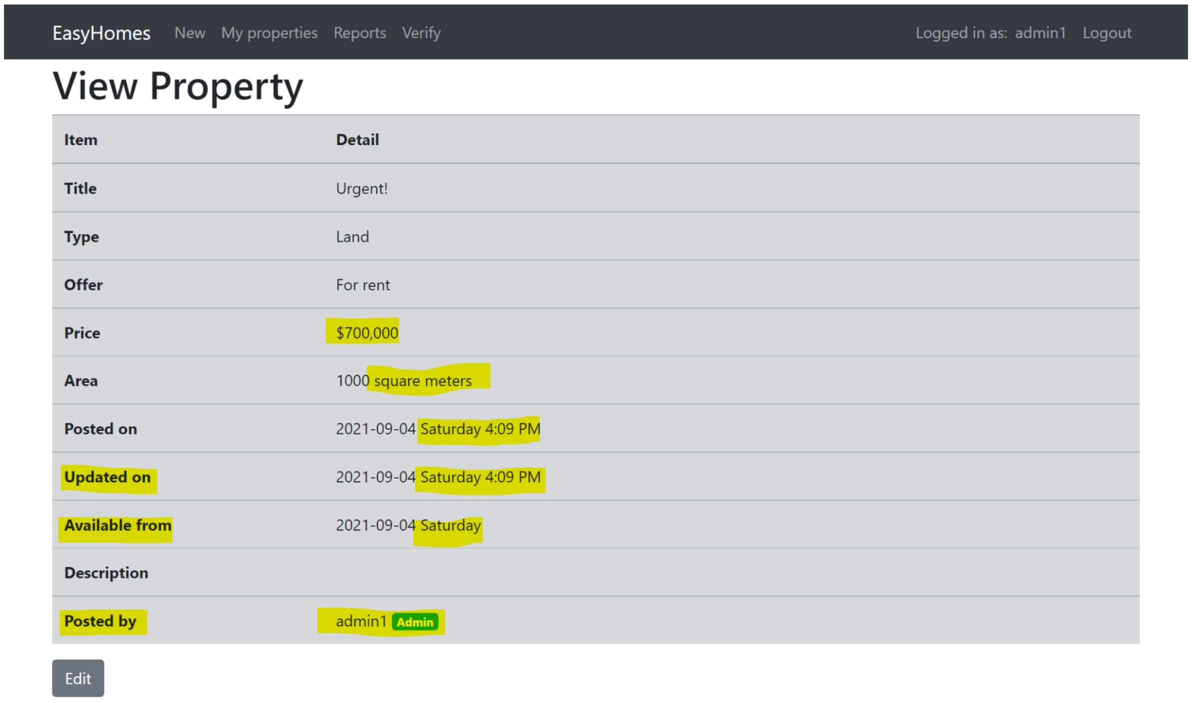
Before



View Property

id	18
createStamp	2021-09-01T13:06:51.013Z
updateStamp	2021-09-01T13:06:51.013Z
title	Local Precinct 420
description	Protect and serve. Our building is for sale, seized catnip and all included
area	200
type	commercial
offer	sale
dateAvailable	2021-09-17T00:00:00.000Z
price	250000
installments	
userId	2a5839c1-6d4c-49f9-846a-c9fab0085aef

After



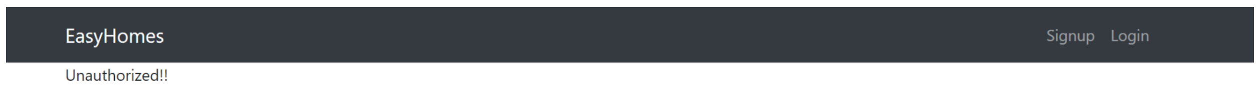
View Property

Item	Detail
Title	Urgent!
Type	Land
Offer	For rent
Price	$700,000
Area	1000 square meters
Posted on	2021-09-04 Saturday 4:09 PM
Updated on	2021-09-04 Saturday 4:09 PM
Available from	2021-09-04 Saturday
Description	
Posted by	admin1 Admin

Unauthorized page

When a user tries to access a page without being authorized to do so, e.g. not authenticated or trying to access admin role pages such as /reports, /verify, etc.

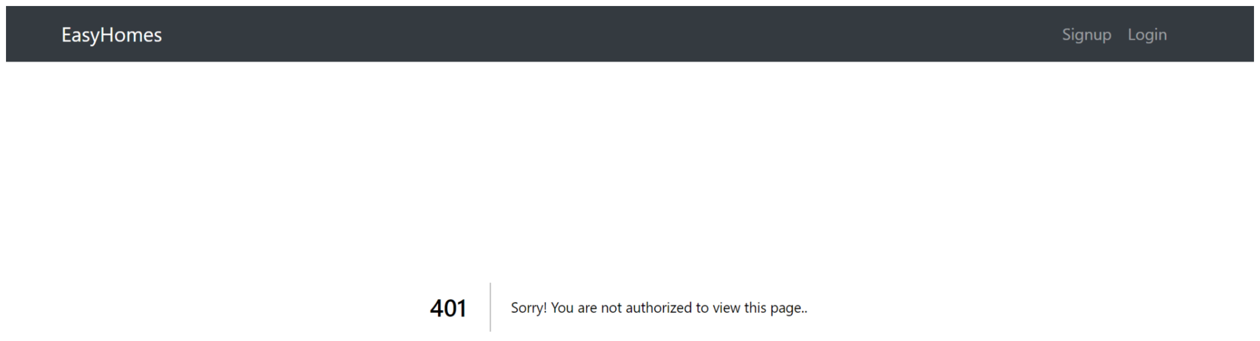
Before



Unauthorized!!

After

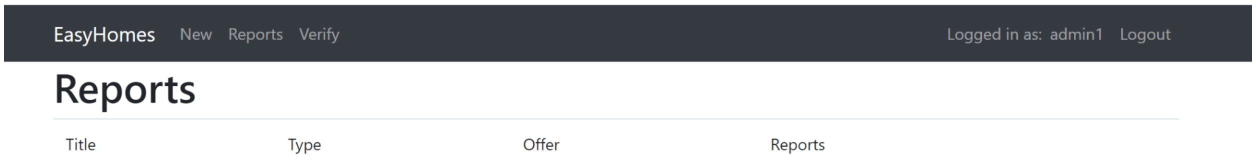
The actual message may change depending on the page being accessed to better inform the user why they are not authorized to view that page.



401 Sorry! You are not authorized to view this page..

Reports page (admin role)

Before

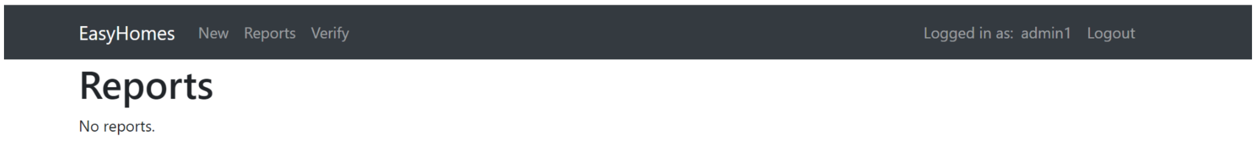


Reports

Title	Type	Offer	Reports

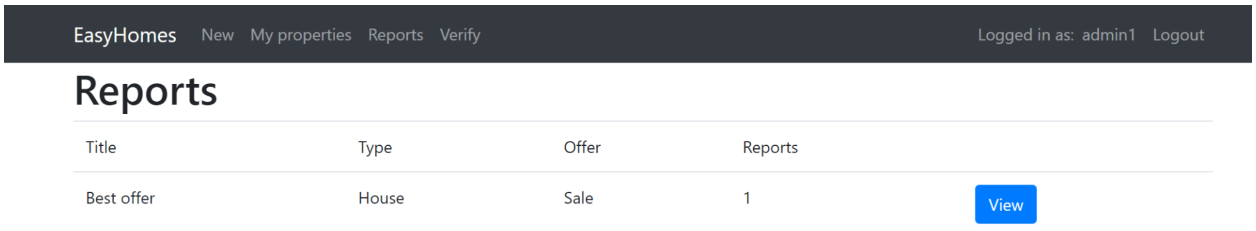
After

Instead of displaying an empty table when there are no reports, we show a message so that the user clearly knows why they can't see any reports.



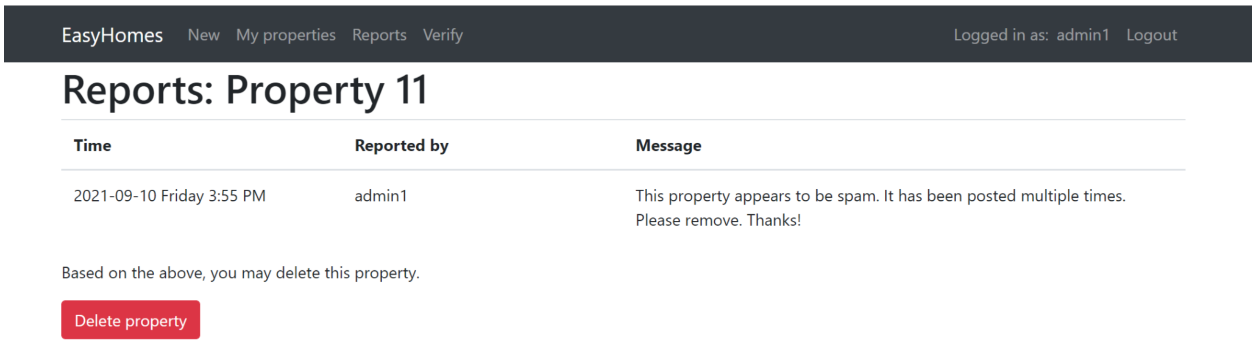
Reports

No reports.



Reports

Title	Type	Offer	Reports	
Best offer	House	Sale	1	View



Reports: Property 11

Time	Reported by	Message
2021-09-10 Friday 3:55 PM	admin1	This property appears to be spam. It has been posted multiple times. Please remove. Thanks!

Based on the above, you may delete this property.

Bibliography

Abeti, 2017. Low Cost House in Kenya? Beware!. [online] Tuko. Available at: <https://www.tuko.co.ke/259521-low-cost-housing-kenya-beware.html> [Accessed 10 May 2021].

Budd, 2007. Heuristics for Modern Web Application Development. [online] Andybudd.com. Available at: <https://andybudd.com/archives/2007/01/heuristics_for_modern_web_application_de> [Accessed 5 August 2021].

Cytonn.com, 2021. Current Real Estate Trends in Kenya & How They Affect Investors. [online] Available at: <https://www.cytonn.com/blog/article/current-real-estate-trends-in-kenya-and-how-they-affect-investors> [Accessed 3 September 2021].

Mwangi, 2020. Most Popular House Hunting Scams to Avoid in Nairobi. [online] Kenyans.co.ke. Available at: <https://www.kenyans.co.ke/news/57809-most-popular-house-hunting-scams-avoid-nairobi> [Accessed 10 May 2021].

Nielsen, J., 2020. 10 Usability Heuristics for User Interface Design. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed 16 August 2021].

Safu, M., 2020. Why you should ask questions to identify fraud in real estate - CommercialKe. [online] CommercialKe. Available at: <https://commercialpropertykenya.com/why-you-should-ask-questions-to-identify-fraud-in-real-estate/> [Accessed 9 July 2021].

Saif Properties, 2021. Property Market Outlook in Kenya 2021– First Half - Saif Properties. [online] Available at: <https://saifproperties.com/property-market-outlook-in-kenya-2021-first-half/> [Accessed 10 August 2021].

Sharp, H., Rogers, Y. and Preece, J., 2019. Interaction design. Indianapolis: John Wiley & Sons.