Core functions:

The core functions of this project are:

1. Properties listings and management
2. A portal for Admins
3. A portal for Landlords
4. A portal for Tenants
5. A portal for Users
6. Book appointments and viewings
7. Users preference profile
8. Visitors tracking system
9. Data analysis system

- The website will let the state agency advertise and market their properties online and reach the wider target audience. This will also allow save the company a lot of time as they wouldn't have to respond to people's queries on the phone regarding any new or upcoming properties, as they will be online and people can just view them in matter of seconds or clicks. This will also give the agency to remove any property that are rented out and sold so people wouldn't have to keep asking if the property is sold or rented yet, as they can just browse the website and look for the status of the properties.

- The admin portal will allow the directors or the website admins to control all the data and users of the website. Through the admin portal they would be able to carry out operations such as the adding, editing, updating, and deleting properties, they would be able to manage all the properties and their statuses, they would be able to manage all the tenants, their documents, their rent schedules, residents living in each property, the amount of debtors, the amount of creditors, add, edit, update and delete landlords, assign landlords with properties, manage the front end content of the website such as add, edit, update and delete news articles, contact information, opening and closing times, the information about the team working in the company, their roles, their social network profiles, they would also be able to add new appointment and property viewing dates and times, approve the requests and also reject them as well as many other extra features.

- The landlord portal will allow all the landlords to manage only their tenants and owned properties. They would also be able to add, edit, update and delete property viewings and appointments, as well as accept and reject the viewings and appointments. Additionally, they will also be able to manage the tenant documents, add, edit and update any entries in the tenants rent schedule so they can keep track of who has paid what and who owes how much. Another main stream function of the landlord portal would be the access to landlords to check the repairs and maintenance logs from their tenants, also allowing them to chat to them regarding the log through the chat system.

- The tenant portal will allow the tenants to view the details about the property they are residing in, they would also be able to view their current lease, their rent schedule in a read only format, they will be able to make payments using PayPal gateway, edit their profile, look at any of their documents that are remotely related to the property they are residing in and also look at digital receipts of all the payments they have made to their landlord or estate agent. Additionally, they will also be able to log any maintenance and repair jobs that need doing in their residential property and also chat to their landlord about the log in using the built in direct chat system in their portal. Nonetheless, they will also be able to book any appointments with the staff working in the company and also view a list of all the booked and rejected appointments and viewings they had requested to be booked.

- The user portal will be designed to make a gateway for better communication between the website visitors and the company, through this portal they will be able to book any appointments with any of the staff working in the company or book any property viewings and also view a list of the outcomes of the bookings. Furthermore, they will also be able to make a personal preference profile.

- The appointment and property viewing system, will allow the users and tenants to book viewings for any upcoming or new residential or commercial properties listed on the website, the tenants and users will also be able to book appointments with any of the staff that are free to provide any advice or consultation about any issues remotely related to real estate, saving the tenants and users the time to contact the office to find their available times to book appointments instead they can just book them from the website which will be making the users and tenants life easy.
- User preference will be a system embedded into the users portal where the users will be able to make a personal preference profile, setting the requirements regards to what kind of property they are hunting for, so every time a new property matching their preferences is uploaded, they get an email with the link to the details page of the property, so they can explore it even further by reading into the description and maybe book a property viewing.

- Visitors tracking system will allow the company to monitor the traffic on their website, by looking at how many people are visiting the website every day in range throughout the year. Furthermore, this will also be a gateway for collecting data for marketing purposes as the visitors tracker system will also tell the company where the traffic is coming form; is it coming from a search engine, any other referring domain or URL or if they traffic is coming using direct access, with what devices and screen resolutions being used to browse the website and which pages were the most browsed, giving them data to analyse and use for further marking and search engine optimisation purposes.

- Data analysis system will be able to provide automatic graphs and charts for the company to analyse using the data collected and stored in the database. This system will produce a bar chart comparing viewings and appointments that took place throughout the year, another pie chart will show how many properties of each category they have which are listed as ‘To Let’ or ‘For Sale’, another chart will show

the current property prices using the purchasing figure of the property and the rate of HPI (House Price Index) over time and show the current guide price of the property, and also the data analysis system will be able to forecast, which category of properties they need more and how many bed rooms properties they should be looking into buying.

Algorithms:

There are various algorithms that will be used in the website, such as:

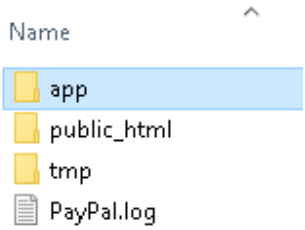
- Appointment booking system:
    - The appointment booking algorithm, will check for all the dates and times that are equal to or greater than today, added by all the employees at the company and will check if there are any available slots to be booked. Once it finds all the slots that are available for booking by all the staff it will then show all those dates and times in the tenant and user portal as a list, to eliminate any confusion when booking and also it will eliminate showing the empty spaces in a calendar. Once the user or tenant finds the perfect time and date for the appointment they request for a slot to be booked, the request is sent to the admin or any of the relevant staff with who the appointment is related, they then accepts or rejects the appointments. If they reject the appointment, a input field has to pop-up for them to provide a reason for rejection, and an email is sent to the user or tenant who requested the appointment, with alternative times available to be booked, else if they accept the appointment the user or tenant gets an email with the confirmation of the appointment with the date and time as a reminder.
- Property viewings booking system
    - The property viewings booking system, will check for all new viewing dates and times added for the new properties listed on the website as ‘To Let’ or ‘For Sale’. This will then show a booking form in all the details page for all the properties listings, which have any dates and times added by the admin, if not, then the form won’t be displayed. However, if there are any slots available for booking then the users/ visitors or the tenants looking to move to a different property can request to book the viewings. The request is then sent to the admin or landlord of the property, they then accepts or rejects the viewing. If they reject the viewing, a input field has to pop-up for them to provide a reason for rejection, and an email is sent to the user/visitor or tenant who requested the viewing, with alternative times available to be booked, else if they accept the viewing request the user/ visitor or tenant gets an email with the confirmation of the viewing with the date and time as a reminder.
- Data validation
    - The data validation algorithm is used in all the input fields of the website, it gets triggered automatically once a user starts typing into an input fields. For instance, if a user want to register to the monthly newsletter they algorithm will detect if the user is entering a real email, by far it can only detect the ‘@’sign and look for a ‘.’ after the ‘@’ sign as of the wide variety of email hosting providers or domains. Another good example of the data validations algorithm would its trigger when someone tries to login to the website, it

checks for the entered username and password and then checks it with the database table 'users' and if an entry matches for instance the username it will output 'wrong password entered' or if a new user wants to register and they type their password and password confirmation and they don't match it will output 'password and confirm password doesn't match!' or if a new users enters a password and it doesn't contain a uppercase letter, a lower case letter or a number the algorithm should make an alert output 'make sure your password contains a uppercase, lowercase and a number!'

- User preferences and property listings match algorithm:
    - This detects all the registered users with a complete personal preference profile on the website and every time a new property is listed as 'To let' or 'For Sale' the algorithm loops through all the preference profile and closely matches the new listings description with the users preferences and check with all the entries they have added for example the post code they want the property in, the minimum rent they are willing to pay, the maximum rent they are willing to pay(as of inflation rent can go up), if they prefer a garden with the property, how many bedroom house they are after, how may bathrooms they are looking for in the property and also if the property landlord allows for keeping pets or no. after taking all such entries and matching them closely with the property description the users with matching preferences get an email to the link of the new property's detail listing page, from where they can read in depth about the property and go forward if they want, booking appointments and viewings if they like it.

Framework:

The website will be made in Object Orientated PHP, with a custom Model View Controller (MVC) framework, as this will allow for defensive programming, preventing from possible SQL Injections and other hacks. The MVC framework is divided into two different folders, the 'app' and the 'public_html'. The 'public_html' folder contains all the stylesheets, images, JavaScript and a config file, with the directory separator linked to the 'app' folder. The 'app' folder contains the 'controllers', 'lib', 'models', 'views', and the 'PayPal SDK'. This is where all the main code of the website is based. When the website is live, the only folder that is visible to the public is the 'public_html' folder, as the 'app' folder is stored outside of the public root directory, so it is placed directly on to the servers root directory and the config file directory separator is used to access the files, so it acts as a firewall against any attacks. With the MVC framework, I can separate all the HTML, CSS, JavaScript, and PHP, in different folders. All the HTML can go in the 'Views' folder, all the PHP can go in the 'Controllers' folder and all the database connections can go in the 'Models' folder.



MVC Break down:

- **Model:** The model represents the data structures. Typically the model classes will contain functions that help retrieve, insert, and update information in the database.
- **View:** The view is the information that is being presented to a user. A view will normally be a web page, but it could also be a page fragment like a header or a footer. It can also be a RSS page, or any other type of page.
- **Controller:** The controller serves as an intermediary between the Model, the Views and any other resources needed to process the HTTP request and generate a web page.

Stylesheet Framework:

The stylesheet framework that I will use for this project will be 'Bootstrap'. The main reason being that it's a widely used framework on the internet with 57% coverage of all sites on the world wide web. Nonetheless, another reason why I would use bootstrap is because I have used it various times before in many projects and also as im very fluent with it and also as it is very easy to use and also makes the websites look very fresh and modern, even though yu have to add some additional custom stylesheet files to manipulate the look of the website to the prototype model asked by the clients or the primary users of the website.

Fonts used in the entire website:

There are various fonts that I will use in the making of this website.

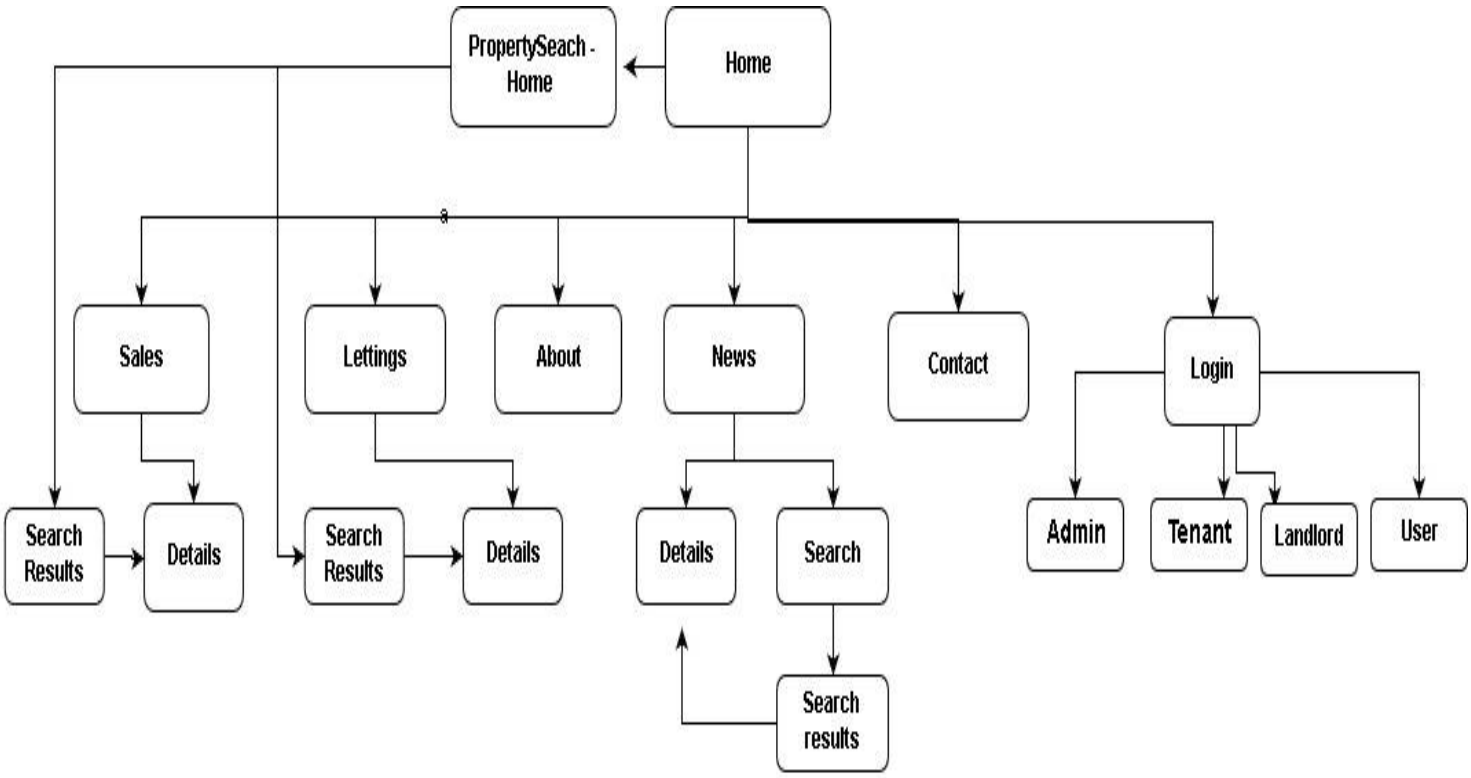
- **HTML:**
  - sans-serif
- **Body- Bootstrap:**
  - "Helvetica Neue", Helvetica, Arial, sans-serif
- **Body:**
  - "Open Sans", sans-serif
- **Navigation:**
  - "Open Sans", sans-serif
- **Newsletter box:**
  - [**h3:** Georgia, "Times New Roman", Times, serif]
  - **[span:** "Open Sans", sans-serif**]**
- **Headings:**
  - "Roboto", sans-serif
- **Body:**
  - "Open Sans", sans-serif
- **Testimonials:**
  - **[heading:** "Roboto", sans-serif**]**
  - **[slider {italic}:** Georgia, "Times New Roman", Times, serif**]**
- **footer:**
  - **[heading:** "Roboto", sans-serif**]**
  - **[links:** "Open Sans", sans-serif **]**

Why google fonts?

The main reason I used google fonts link instead of downloading the fonts to my server is beause by using the link I am eliminating the extra storage space on the hosting server and also deccreasing the website load time as the browsers would have mostly cache the fonts once used from other websites using the same link, as google fonts hosting link is a widely used link on the internet, visitors browser would have the fonts chached so it would make it quick to for the website to load and process leaving the users with a better experience.

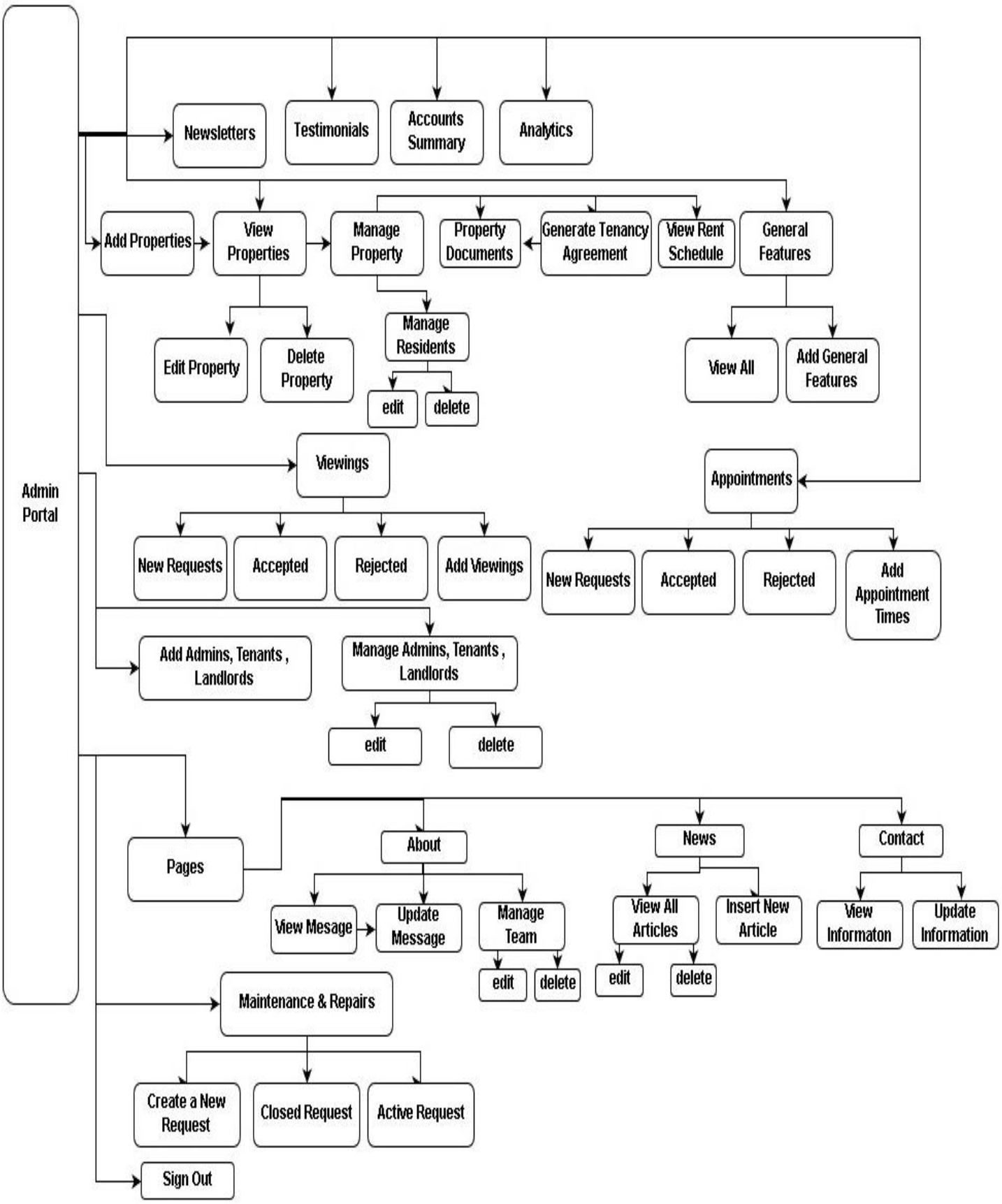
Navigation Structures:

1. Website navigation structure – Front-end:

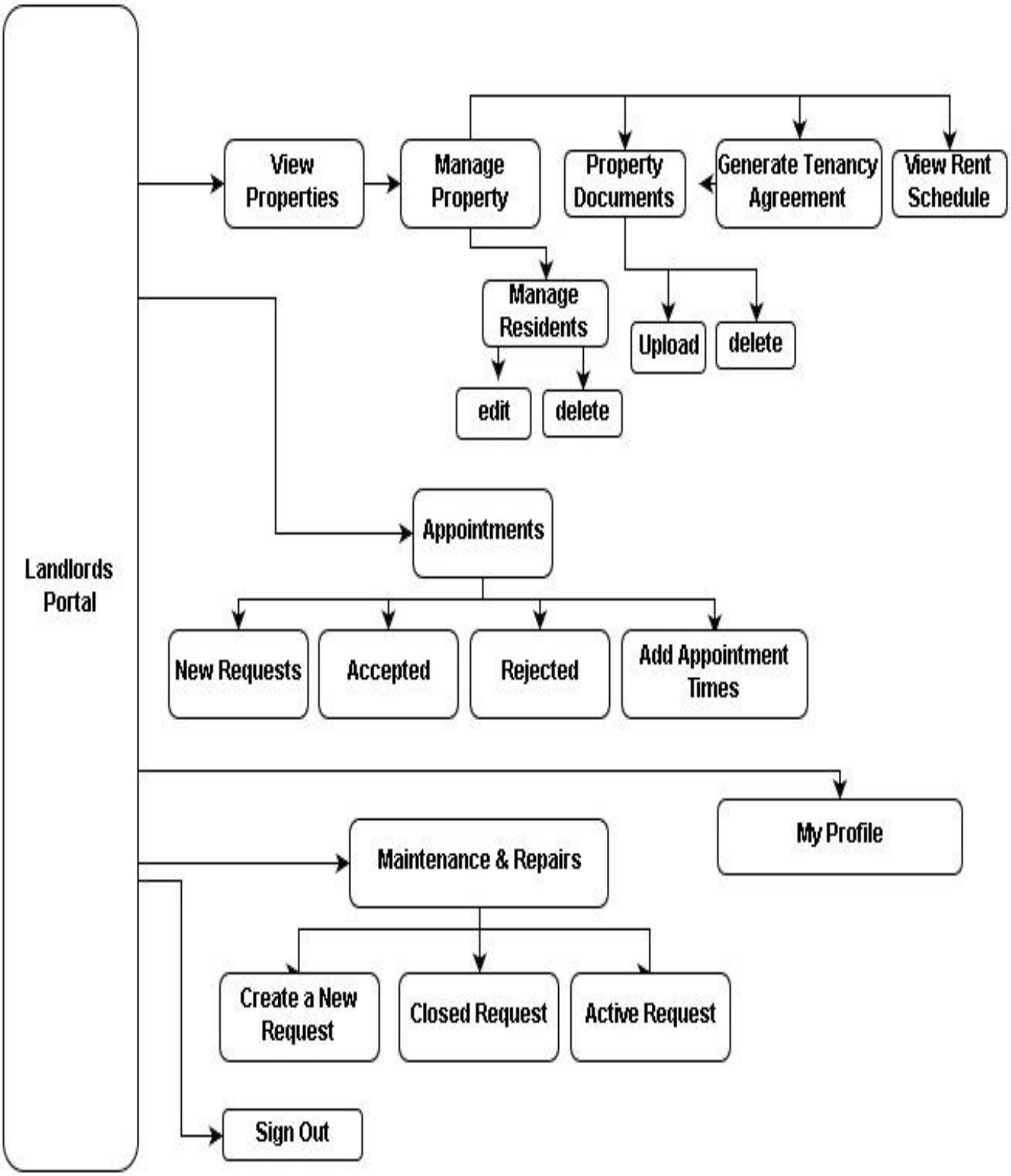


The website is linked together with several pages, each having the ability to navigate back to the previous and related sub-pages. The home page is set as the 'Index' page, with all other pages linked, whenever a page is opened the page name gets underlined in colour green as a reference, indicating active page. The Home page image slider contains a search form where users can search for any properties that are for sale or rent according to the prices they can afford to pay or their preferred location, the 'PropertySearch – Home' as indicated in the diagram above will redirect the users to the 'sales.php' or the 'lettings.php' according to their search results.

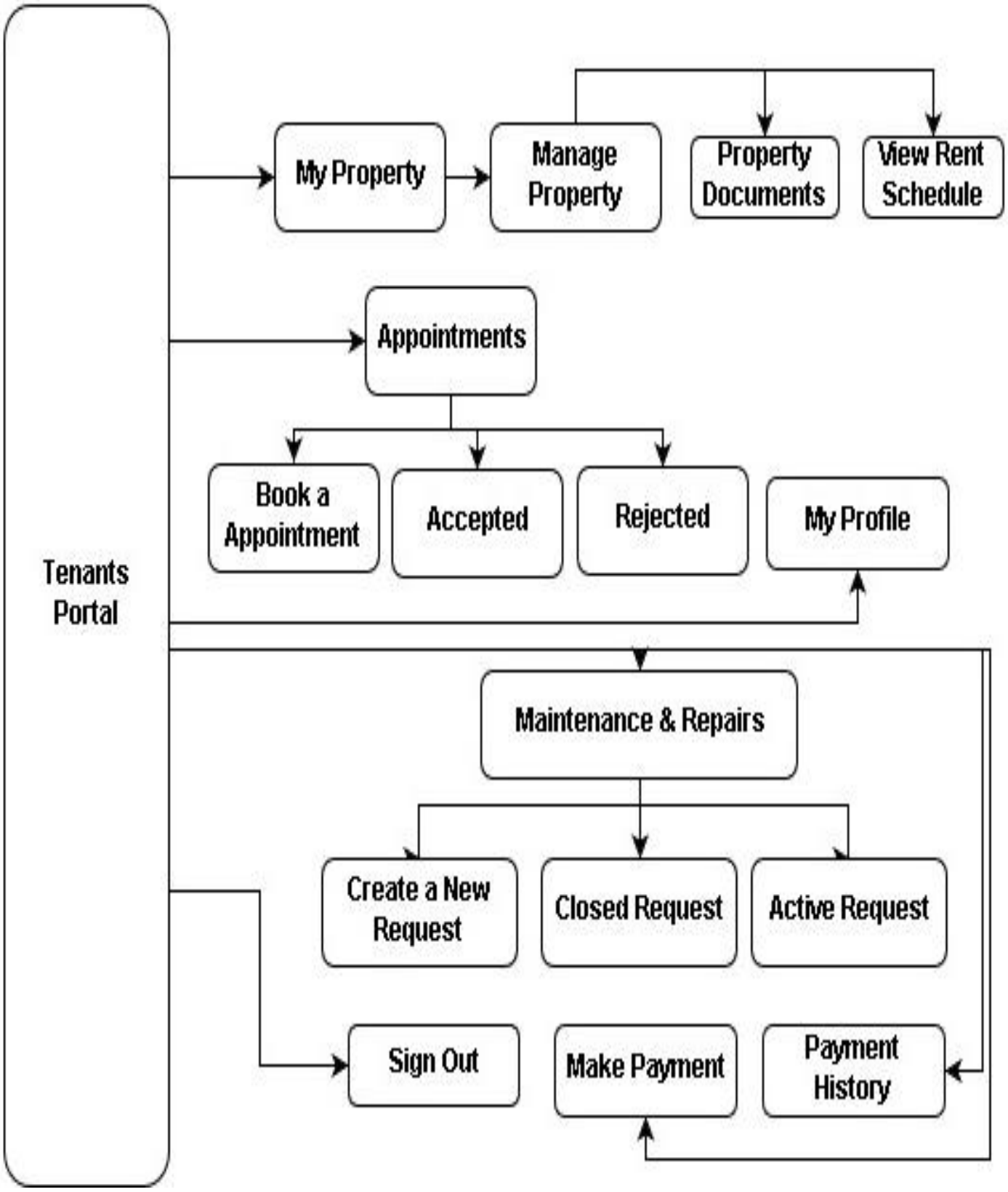
2. Website navigation structure – Admin Panel:



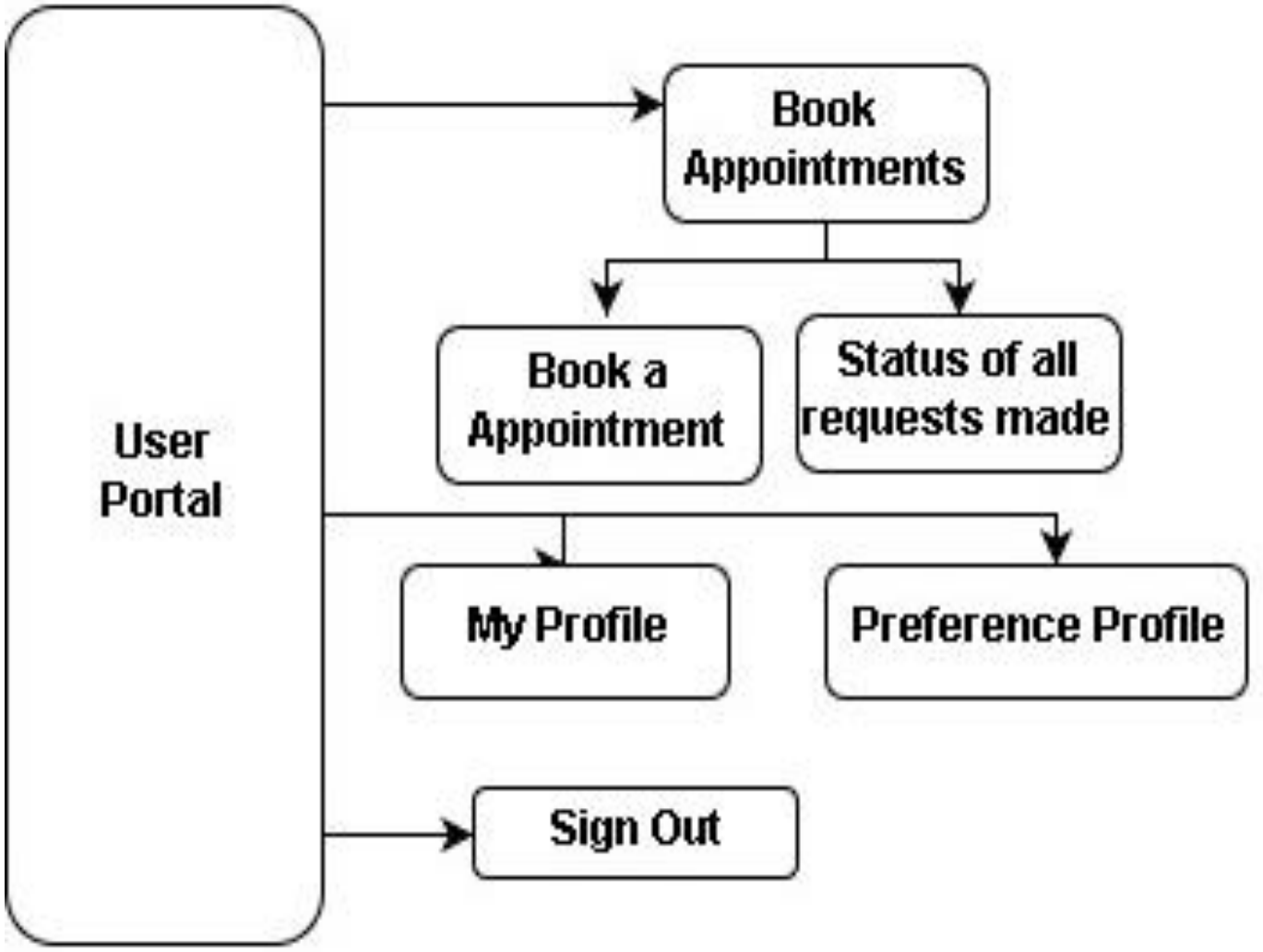
3. Website navigation structure – Landlord Panel:



4. Website navigation structure – Tenant Panel:

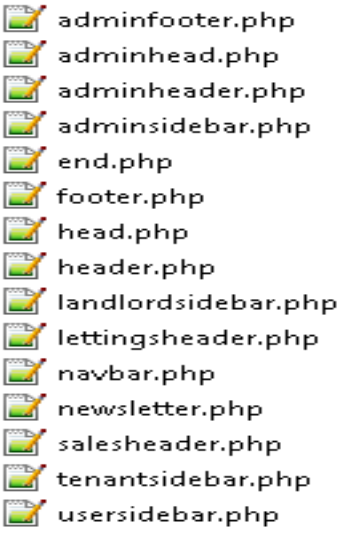


5. Website navigation structure – Users Panel:



Design Template:

The website design for all the front-end till the back end is divided into a sections and then included into other pages, these template pages are all stored in the **_template** folder. This includes the footers for all the panels in the back end, the navigation sidebars, headers, it also contains all the templated pages for the front-end such as the head (to store the Meta data), the header, and the newsletter column, the navigation bar, the footer as well as the sales and lettings pages header.



Why I chose to use a template structure?

The reason why I chose to make a template structure for the website, is because that would save me a lot of time programming the same thing again and again for all the pages, and also as this is a very big project, having a template structure would aid me in modification, as if later in the project or in the future if I have to add or modify some information in any of the section of the website I can simply go to its template file and change from there instead of going through every single page on the website and replicating the steps. Furthermore, I believe this is more professional way of programming then replicating data and making the code look very chaotic.

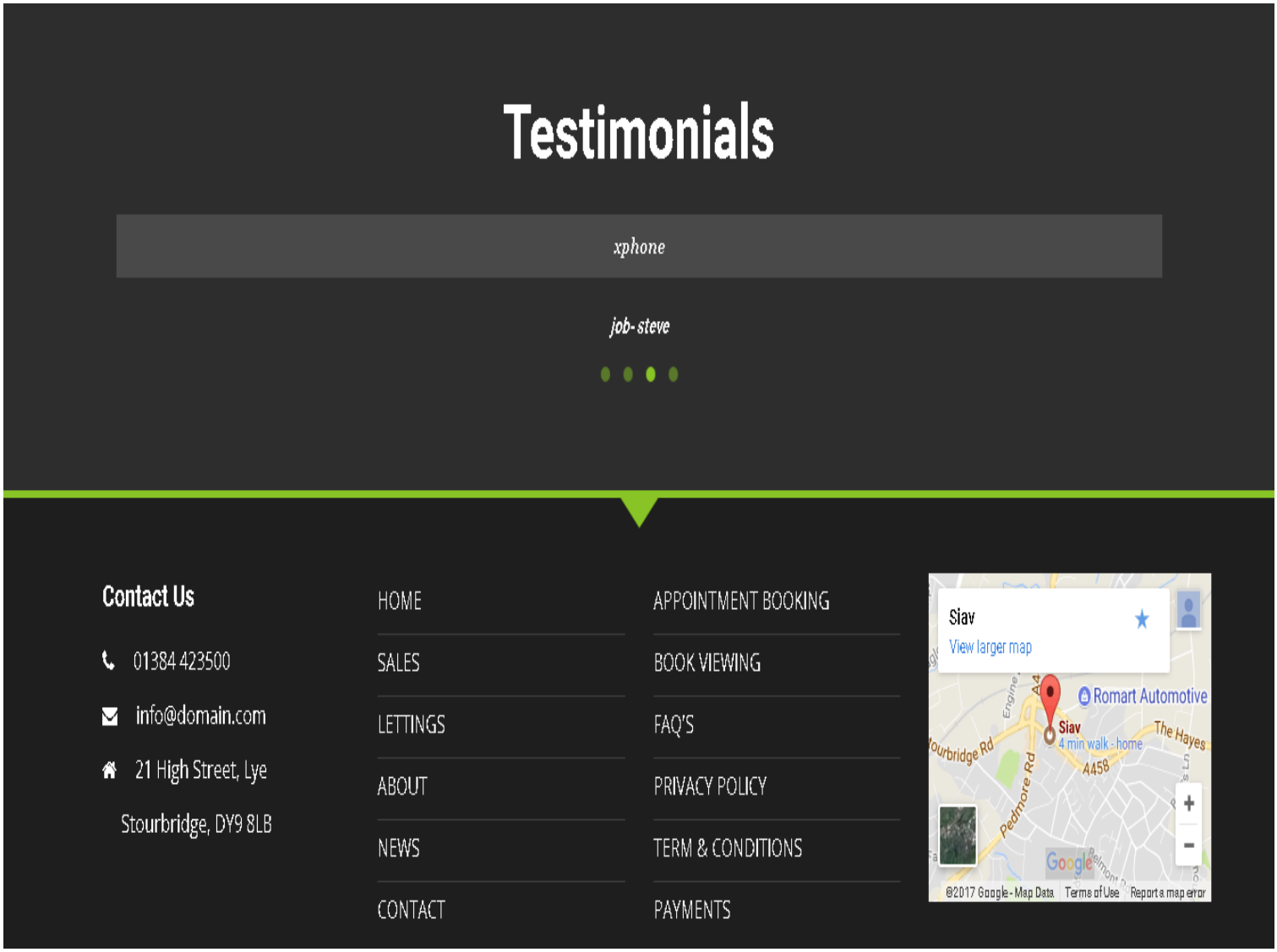
Front-end Template:

- The navigation bar:
    - The navigation bar is a template on its own, which is then included in all other pages in the front-end. It's linked with all other pages so every time a page from the navigation bar is opened it gets marked as active and gets underlined with the colour green. This would be beneficial for the web developer later in the future if they decide to add or manipulate any information on the website if necessary.

- The newsletter column:
    - The newsletter column is also a template on its own as well, it is set to a width of 'col-md-12' simplified as 100%, meaning it could be included on any page and also it would adopt the same functionality as all other pages and would make the code more organised.

- The Footer end:
    - The footer end is kept simple with all copyright information and social icons and is included in all other pages.

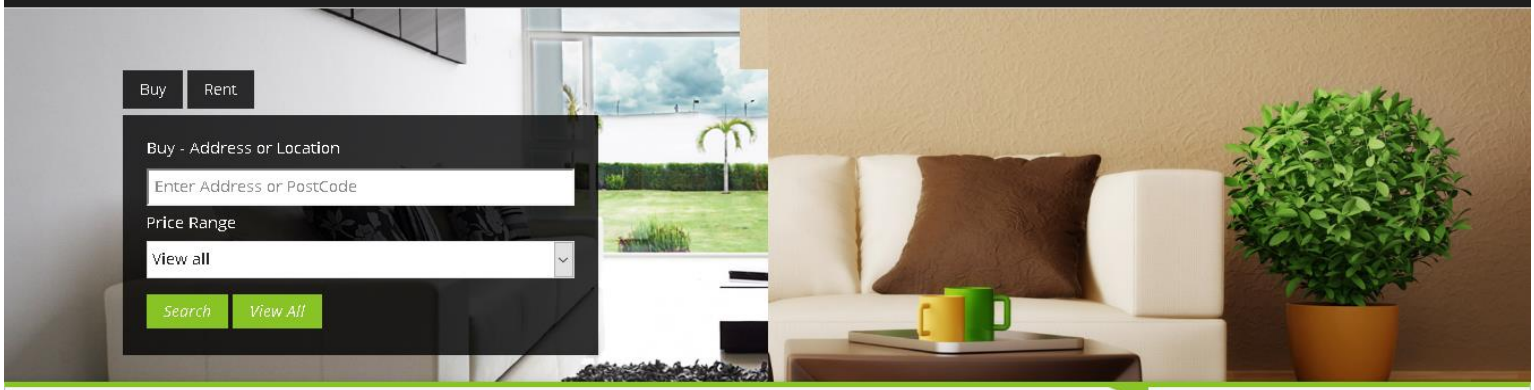
- The Footer Middle:
    - The footer middle contains a testimonial slider and a low 4 columns footer, one column with company details 2 columns with links to other pages and one column with an interactive google map so people can see where the real estate agency actually is.



- **<u>Front end full page design example:</u>**

SIAV

HOME SALES LETTINGS ABOUT NEWS CONTACT LOGIN



STAY INFORMED - *Subscribe to get info about upcoming properties right to your inbox.*

Your Email Address

FEATURED PROPERTIES

Property	feat test
Beds	5
Baths	7
Price	£400

Property	Tree House
Beds	2
Baths	1
Price	£600

Property	4a Hill Road
Beds	85
Baths	56
Price	£55200

Testimonials

xphone

job-steve

Contact Us

01384 423500

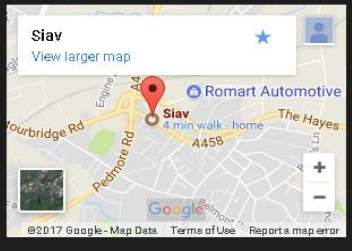
info@domain.com

21 High Street, Lye

Stourbridge, DY9 8LB

HOME
SALES
LETTINGS
ABOUT
NEWS
CONTACT

APPOINTMENT BOOKING
BOOK VIEWING
FAQ'S
PRIVACY POLICY
TERM & CONDITIONS
PAYMENTS

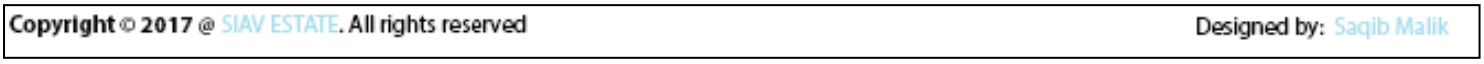


Back-end Portal Template:

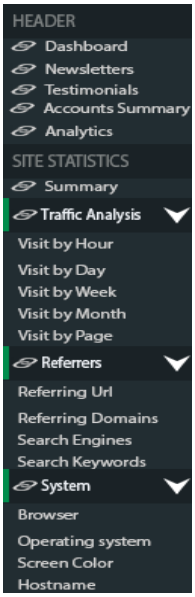
All the portals in the back end of the website such as the admin, landlord, tenant and use portal, inherit the same design so it was more convenient for me to design a template which they all can inherit and then it can be manipulated accordingly to the navigation structure. In the back end the template is divided into header, sidebar and footer. So these are they template designs of the back end portals:

- The header:
    - The header contains all the login activated profile dropdown views, the navigation collapse/ toggle button and the logo.

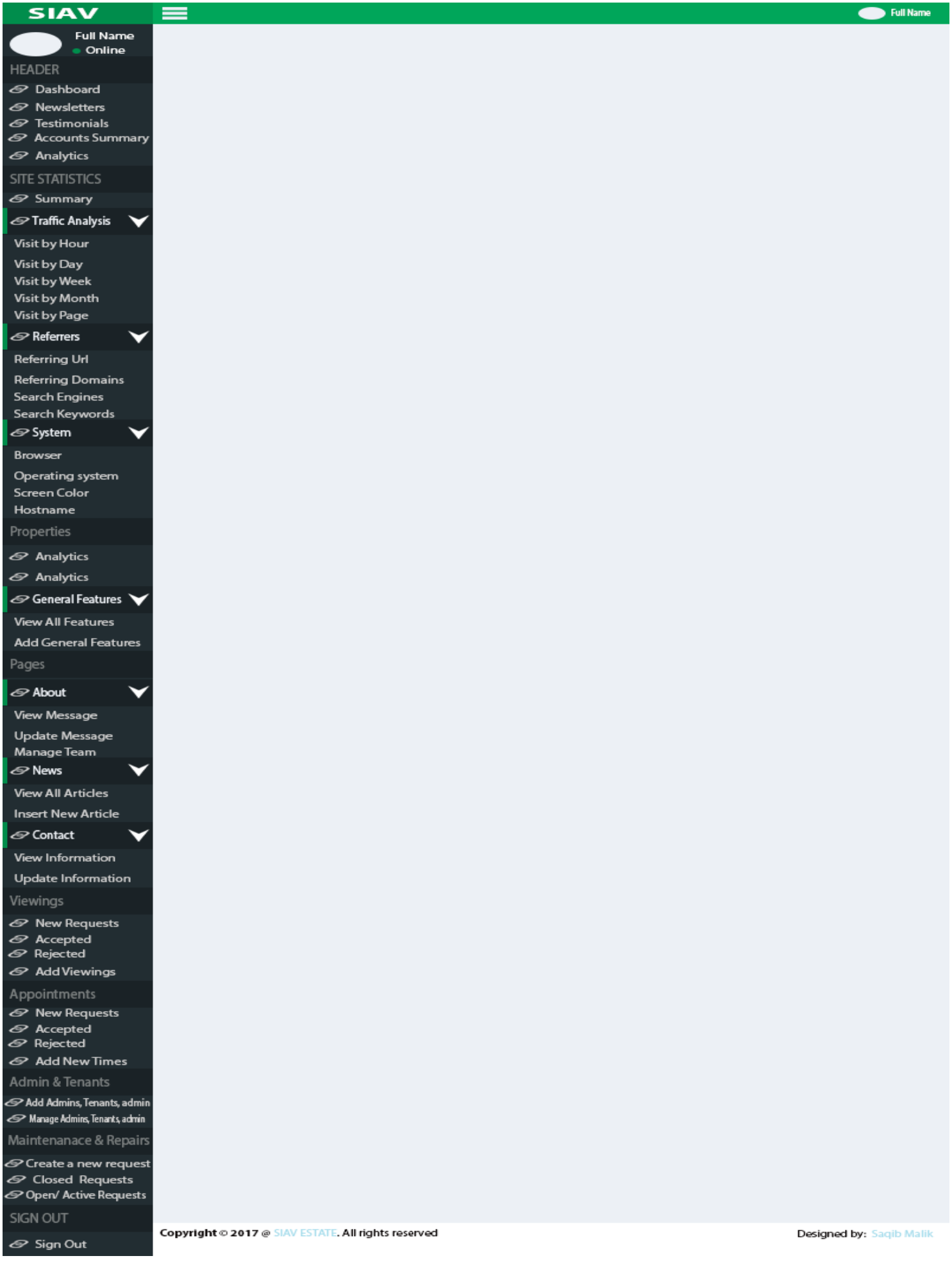
- The Footer:
    - The footer contains all the copyright information and also the web developer's link to their website or any other social profile.



- Sidebar example design:
    - The following side bar template design was the initial template design for the admin panel, which has then been manipulated in all other panels, to fit the navigation structure of that particular portal. The initial design of the admin panel only contains some pages as it's just a prototype and also if all pages were included in the design section then it would be too long and too time consuming, where I could just extend it quickly whilst programming. Additionally the side bar contains all the links that are defined in the navigation structure for the admin panel and all other panels.



- Full page design of the back end portals:
    - This full page design contains the navigation structure of the admin panel.


SIAV
Full Name
Full Name
Online
HEADER
Dashboard
Newsletters
Testimonials
Accounts Summary
Analytics
SITE STATISTICS
Summary
Traffic Analysis
Visit by Hour
Visit by Day
Visit by Week
Visit by Month
Visit by Page
Referrers
Referring Url
Referring Domains
Search Engines
Search Keywords
System
Browser
Operating system
Screen Color
Hostname
Properties
Analytics
Analytics
General Features
View All Features
Add General Features
Pages
About
View Message
Update Message
Manage Team
News
View All Articles
Insert New Article
Contact
View Information
Update Information
Viewings
New Requests
Accepted
Rejected
Add Viewings
Appointments
New Requests
Accepted
Rejected
Add New Times
Admin & Tenants
Add Admins, Tenants, admin
Manage Admins, Tenants, admin
Maintenanace & Repairs
Create a new request
Closed Requests
Open/ Active Requests
SIGN OUT
Sign Out
Copyright © 2017 @ SIAV ESTATE. All rights reserved
Designed by: Saqib Malik

Website Validation, Sanitisation, Security and Integrity of Data

Data protection:

All companies or websites that store user data have to comply with the data protection act, which is **Data Protection Act 1998,** meaning they have a responsibility to protect the data from being hacked from their databases and servers. So for this particular reason, I have to make sure I eliminate all possible ways that someone can hack into my website or server and also I have to make sure hackers aren't able to SQL Inject the website or run any XXS attacks, and also aren't able to launch any brute force attacks on the website to crack any passwords.

So taking all these vulnerabilities into account I have used some strong and power algorithms some of which are premade or inbuilt PHP algorithms and some which I made myself.

Sanitising form inputs

Nowadays its very easy to run XXS attacks on website as many website devlopers don't think of the need to sanitise the user inputs, however, this is one of the most vital part of web developemnt, as if the user inputs aren't sanitised then the users can just add any sort of scripts in the input fields to manipulate the queries and fetch any unautorised data. So to prevent all of the unauthorised inputs and invalid data I will run a custom made data validation algoritm and will also use the in built **htmlentities()** function.

htmlentities

(PHP 4, PHP 5, PHP 7)
htmlentities — Convert all applicable characters to HTML entities

Description

```
string htmlentities ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string
$encoding = ini_get("default_charset") [, bool $double_encode = TRUE ]]] )
```

This function is identical to htmlspecialchars() in all ways, except with **htmlentities()**, all characters which have HTML character entity equivalents are translated into these entities.

If you want to decode instead (the reverse) you can use html_entity_decode().

The PHP function htmlentities() is used to convert any special character that are inserted or typed into a input field in a HTML form to HTML character entities, so this would result in the prevention of the given input string containing any special characters, that can cause the inputs to cause data manipulation while sending requests to the database as queries or URL's. For instance, if we have single quotes with an input string which is required to be embedded with a query, will cause PHP error due to the incompletion of the query statement, which is truncated by the single quotes. In such situation, htmlentities() are used to prevent parsing special characters occurred with user input data.

Data validation custom algorithm:

- Different keys are defined for different types of inputs, for example an input type email will have a different key to an input type phone number or zip code.
- Keys are stored as arrays.
- Example of key defined for the input type date would be as follow:
    - "^[0-9]{4}[-/][0-9]{1,2}[-/][0-9]{1,2}\$",
- So this key will check if the date is in the form:
    - Year (- | /) Month (- | /) Date
    - This checks if the year contains 4 digits, the month contains 2 digits and the date also contains 2 digits.
    - This also checks if the month, date and year are separated with specific characters or not, the characters allowed are a (–) sign or a (/).
- The validation array is set to failure I equal to false.
- The algorithm then loops through the forms and checks for each input(item) as key, if the input fields data has a value greater than 0, if zero then check for mandatory fields if the field if mandatory then output, required else check if the input matches the key. If the key matches the data input then continue else result in an output alert ('some of the data entered is invalid!').
- Then the data validation form sanitises single variables to types, allowing for static calling to allowing simple sanitisation.
- This then runs cases if the case is 'url' then run the inbuilt FILTER_SANITIZE_URL function, if case is integer then run the FILTER_SANITIZE_NUMBER_INT function etc.
- Then the algorithm will return the data in sanities form and return to the queries and URLs preventing for any XXS attacks.

Brute Force attack prevention:

- Brute force attacks are prevented in the login systems by adding a key to it, which defines the password while registering should contain at least one upper case letter, one integer, one special character and greater than or equal to and less than 255 characters, lower case letters, or its gets very complicated to brute force and can even take millions and billions of years to cracks as of the complexity of the passwords and the possible combinations to be tried, to crack.

Elimination of global variables:

- To make the site more secure, I will eliminated the use of the global variables, through out, whilst programming, this will be done through the use of inheritance where I will be using, parent ➔ child inheritance, and this will allow the connections to inherit all the public and private variables throughout the program. Furthermore, this will also eliminate the duplication of code and also will promote reusability and would help in better abstraction of data therefore leading in better design.

Database structure

Properties table:

#	Name	Type	Null	Default	Extra
1	property_id	Int(10)	No	*None*	AUTO_INCREMENT
2	cat_id	Int(10)	No	*None*	
3	date	Int(10)	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
4	title	text	No	*None*	
5	image	text	No	*None*	
6	beds	Int(10)	No	*None*	
7	baths	Int(10)	No	*None*	
8	place	text	No	*None*	
9	price	Int(10)	No	*None*	
10	purchaseprice	Int(11)	No	*None*	
11	updated_price	Int(11)	No	*None*	
12	last_updated	date	No	*None*	
13	description	medium text	No	*None*	
14	keywords	text	No	*None*	
15	features	text	No	*None*	
16	status	text	No	*None*	
17	landlord_id	Int(10)	No	*None*	

This table stores all the information regard to the properties that are being listed on the website and also those that are rented out, along with info regards to the 'landlords_id' being the pragma to link to the landlords table. To store the landlord for each property.

Users Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	fulllname	text	No	None	
3	username	text	No	None	
4	email	Varchar(100)	No	None	
5	password	Varchar(50)	No	None	
6	phone	Int(20)	No	None	
7	role	Text	No	None	

8	title	Varchar(50)	No	None	
9	image	blob	No	None	
10	create_date	datetime	No	CURRENT_TIMESTAMP	

This table will store all the information about the users of the website who have registered an account, this includes the admins, tenants, landlords and ordinary users.

Testimonials Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	review	text	No	None	
3	name	text	No	None	
4	occupation	text	No	None	

This table will store all the testimonials that will be displayed on the front end of the website, for the new potential tenant's perusal. This table will also hold the reviewer's information such as their name and occupation, so the people can judge the credibility of the review.

Terms and Conditions Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	content	longtext	No	None	

This table will store the content of the Terms and Conditions page, and the reason why a separate table is made as this way I can Isolate the information and also use the separate entries in the database to extract the content as an array to make a divisible tabs, so it looks more professional as well and also as this would keep the database table clean.

Tenants Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	user_id	Int(11)	No	None	
3	property_id	Int(11)	No	None	
4	resident_from	Varchar(100)	No	None	
5	resident_to	Varchar(100)	No	None	
6	lease_term	Int(11)	No	None	
7	contract_renewal	date	No	None	
8	minimum_term	Int(11)	No	None	
9	maximum_term	Int(11)	No	None	
10	furnishing	text	No	None	

This table will store all the information about the tenant's such as their lease term, property contract renewal date their user_id linked with the id from the users table for cross reference.

Team Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	name	text	No	None	
3	Image	text	No	None	
4	occupation	text	No	None	
5	facebook	Varchar(200)	No	None	
6	twitter	Varchar(200)	No	None	
7	googleplus	Varchar(200)	No	None	

This table will store all the information about the team working in the company. The table will hold their personal information and their social information such as their Facebook, Twitter and Google Plus profile links.

Subscription Form Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	email	text	No	None	
3	date_susbscribed	text	No	None	
4	date_unsubscribed	text	No	None	
5	Status	Varchar(200)	No	None	

This form hold the email addresses of the all people who have subscribed to the websites newsletter and also stores the date they subscribed and unsubscribed and status. The status is kept as 'T' (Tue) or 'F' (False), where true alerts the newsletter algorithm to send it to this email whereas F tells it no to send, and if the status is F and date unsubscribed is not null and the user re registers the date unsubscribed gets null and subscribed gets to current date stamp and status gets over ridden to 'T'.

Services Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	property_id	Int(11)	No	None	
3	requested_by	Int(11)	No	None	
4	title	text	No	None	
5	description	text	No	None	
6	status	Varchar(50)	No	None	
7	date	date	No	None	

8	last_updated	Varhar(50)	No	None	
9	open_closed	Text	No	None	
10	priority	Text	No	None	
11	assigned_to	Int(11)	No	None	

This table stores all the maintenance and repair logs that have been logged by the tenants for their landlords perusal, so they can get notified of the issues and get them quickly assigned to someone and also be used for communication between the tenant and the admin as cross referenced with the discussions table.

Residents Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	property_id	Int(11)	No	None	
3	name	text	No	None	
4	relation	text	No	None	

This table stores the resident's details of all the residents living in a property with a tenant and their relationship with the tenant so in any case an incident or anything happens they can be approached as well as the tenant

Request Viewings Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	avail_viewing_id	Int(11)	No	None	
3	property_id	Int(11)	No	None	
4	user_id	Int(11)	No	None	
5	date	date	No	None	
6	qty	Int(11)	No	None	
7	Status	Int(11)	No	0	

Request Appointments Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	None	AUTO_INCREMENT
2	avail_appointments_id	Int(11)	No	None	
4	user_id	Int(11)	No	None	
5	appointment_with	Int(11)	No	None	
6	status	Int(11)	No	0	
7	date	date	No	None	

Rent Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	user_id	Int(11)	No	None	
4	landlord_id	Int(11)	No	None	
5	property_id	Int(11)	No	None	
6	pay_date	Varchar(20)	No	None	
7	period	Int(11)	No	None	
8	rent	Int(11)	No	None	
9	amount_paid	Int(11)	No	None	
10	last_updated	Varchar(50)	No	None	

This table stores all the dates and amounts the tenants have made their rent payments and to which period the payment relates to.

Privileges Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	user_id	Int(11)	No	None	
3	tenants_admin	Int(1)	No	None	
4	analytics	Int(1)	No	None	
5	properties	Int(1)	No	None	
6	service_requests	Int(1)	No	None	
7	viewings	Int(1)	No	None	
8	appointments	Int(1)	No	None	
9	payments	Int(1)	No	None	
10	site_content	Int(1)	No	None	
11	site_news	Int(1)	No	None	
12	site_settings	Int(1)	No	None	

The privilages table is made for the admins, to assign new website admins with certain or all privilages, of what pages they can or can’t access.

Preferences Table:

#	Name	Type	Null	Default	Extra
1	id	Int(11)	No	None	AUTO_INCREMENT
2	user_id	Int(11)	No	None	
3	family_size	Int(11)	No	None	
4	Location	text	No	None	
5	Bedrooms	Varchar(20)	No	None	
6	bathrooms	Int(11)	No	None	
7	garden	Int(1)	No	None	
8	Pets	Int(1)	No	None	
9	Min_rent	Int(11)	No	None	
10	Max_rent	Int(11)	No	None	

This table is linked with the users table with the 'user_id', and is used to store the preferences of the users, which they set in their preference profile from the tenant portal, so when a new property matching their preferences is uploaded they get notified of it through email.

Properties Table:

#	Name	Type	Null	Default	Extra
1	Property_id	int(10)	No	None	
2	Cat_id	int(10)	No	None	
3	Date	timestamp	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
4	Title	text	No	None	
5	Image	Text	No	None	
6	Beds	Int(11)	No	None	
7	Baths	Int(11)	No	None	
8	Place	Text	No	None	
9	Price	Int(11)	No	None	
10	purchaseprice	Int(11)	No	None	
11	Updated_price	Int(11)	No	None	
12	Last_updated	Date	No	None	
13	Description	Mediumtext	No	None	
14	Keywords	Text	No	None	
15	Features	Text	No	None	
16	Status	Text	No	None	
17	landlords_id	Int(11)	No	None	

This table store all the information about the properties that will be listed on the website either for display or those sold or rented out. It will hold all crucial information of the properties such as the landlord of the property, the price at which the property was bought and also the date the purchase price was last updated using the house price index value and also how much it increased by. This table will also store the status of the property whether it is listen as for sale or to le, inactive or rented.

Payments_due Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	None	AUTO_INCREMENT
2	Rent_id	Int(11)	No	None	
3	Date	Date	No	None	
4	Amount	Int(11)	No	None	

This table will store all the payments that are due and unpaid by the tenants and will act as a gateway (interlink) with the rent table, linking to each property and tenant.

Payments Table:

#	Name	Type	Null	Default	Extra

1	Id	Int(11)	No	None	AUTO_INCREMENT
2	User_id	Int(11)	No	None	
3	Property_id	Int(11)	No	None	
4	Landlord_id	Int(11)	No	None	
5	Amount	Varchar(10)	No	None	
6	Date	Date	No	None	
7	Payment_type	Text	No	None	
8	Payment_for	text	No	None	

This table will store all the payments that are made by tenants and will also store who the payment was made to and for what and what was the payment type such as, expenses payment or rent deposit, linking back to the landlords (users) and property table.

News_letter Table:

#	Name	Type	Null	Default	Extra
1	id	int(11)	No	None	AUTO_INCREMENT
2	Documents	Varchar(10)	No	None	
3	Name	Text	No	None	
4	Sent_to	Varchar(50)	No	None	
5	Sent_by	text	No	None	
6	Date	Varchar(50)	No	None	

This table will store all the Meta data of the newsletters that have been sent out, including the date, the name of the document, the group the newsletter is sent to for example the admins, users, tenants or landlords. It also store who the newsletter is sent out by, if it's the admin, or landlords.

Housevalbyarea Table:

#	Name	Type	Null	Default	Extra
1	id	int(11)	No	None	AUTO_INCREMENT
2	County	text	No	None	
3	percentage	Varchar(11)	No	None	

This table will store the current house price index of all the counties where the company owns properties in, and will the store the current HPI, so it can be used to calculate the new guide price of the properties.

Guarantor Table:

#	Name	Type	Null	Default	Extra
1	id	int(11)	No	None	AUTO_INCREMENT
2	Property_id	int(11)	No	None	
3	name	text	No	None	
4	Email	Varchar(11)	No	None	
5	Address	Varchar(11)	No	None	
6	Address2	Varchar(11)	No	None	

7	Address3	Varchar(11)	No	None	
8	City	text	No	None	
9	State	text	No	None	
10	County	text	No	None	
11	Post_code	int(11)	No	None	

This table will store information about the guarantor of each tenant, including their residential address so they can be approached or chased in times the tenant can can't be approached or pay up.

Feat (Features) Table:

#	Name	Type	Null	Default	Extra
1	ID	Int(11)	No	None	AUTO_INCREMENT
2	Feat	Text	No	None	
3	status	Text	No	None	

This table will store all the general features properties are likely to have an when inserting a new property listing this should give them the a list as a radio button input forms they can just tick off the features the property has so they are displayed on the front-end of the website when the property is listed.

Expenses Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	None	AUTO_INCREMENT
2	Service_id	Int(11)	No	None	
3	Name	Text	No	None	
4	Vendor	Text	No	None	
5	Date	Date	No	None	
6	Amount	Varchar(10)	No	None	
7	Description	Text	No	None	
8	notes	text	No	None	

This table will store all the company expenses related to the properties, including all expenses related to the maintenance and repairs of the properties.

Documents Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	None	AUTO_INCREMENT
2	Property_id	Int(11)	No	None	
3	Document	Varchar(15)	No	None	
4	Name	Text	No	None	
5	Description	Text	No	None	
6	Date	Datetime	No	CURRENT_TIMESTAMP	
7	Uploaded_by	Int(11)	No	None	

This table will hold all the information about the documents related to each property and tenant and who the document is uploaded by, the types of documents that will be stored are tenancy agreement and any other proof of previous address letters etc.

Discussions Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	*None*	AUTO_INCREMENT
2	Service_id	Int(11)	No	*None*	
3	Sender	Int(11)	No	*None*	
4	Message	Text	No	*None*	
5	date	Datetime	No	CURRENT_TIMESTAMP	

This table will store all the discussions / chat system entries, regarding all the maintenance and repairs between the tenant and landlord.

Categories Table:

#	Name	Type	Null	Default	Extra
1	Cat_id	Int(11)	No	*None*	AUTO_INCREMENT
2	Cat_title	Text	No	*None*	

This table will store all the categories names of all the possible categories a property can have and when a new property is being listed a dropdown menu would be inserted to allow the admin to pick the property related category so front-end users can filter the properties.

Avail_viewings_dates Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	*None*	AUTO_INCREMENT
2	Avail_viewings_id	Int(11)	No	*None*	
3	Time	Varchar(50)	No	*None*	
4	Day	Varchar(2)	No	*None*	
5	Month	Varchar(10)	No	*None*	
6	year	Int(4)	No	*None*	

This table will store all the times, day, month and year of each viewing listing as the rest of the information is stored in the avail_viewings table so data is kept organised and is accessed by the use of a foreign key.

Avail_viewings Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	*None*	AUTO_INCREMENT
2	landlord_id	Int(11)	No	*None*	
3	Date	Varchar(50)	No	*None*	
4	Time	Varchar(50)	No	*None*	
5	duration	Varchar(50)	No	*None*	

This table will store all the information some of the information about the viewings such as the date time and duration, as it's a link table between the avail_viewings_dates table, linking to specific landlord.

Articles Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	*None*	AUTO_INCREMENT
2	Title	text	No	*None*	
3	Author	Varchar(50)	No	*None*	
4	Date	date	No	*None*	
5	Image	text	No	*None*	
7	Keywords	text	No	*None*	
8	content	text	No	*None*	

This table will store the information about all the news remotely related to the company. The table will hold information such as the author of the article, date uploaded, keywords so people can search the article and article related image.

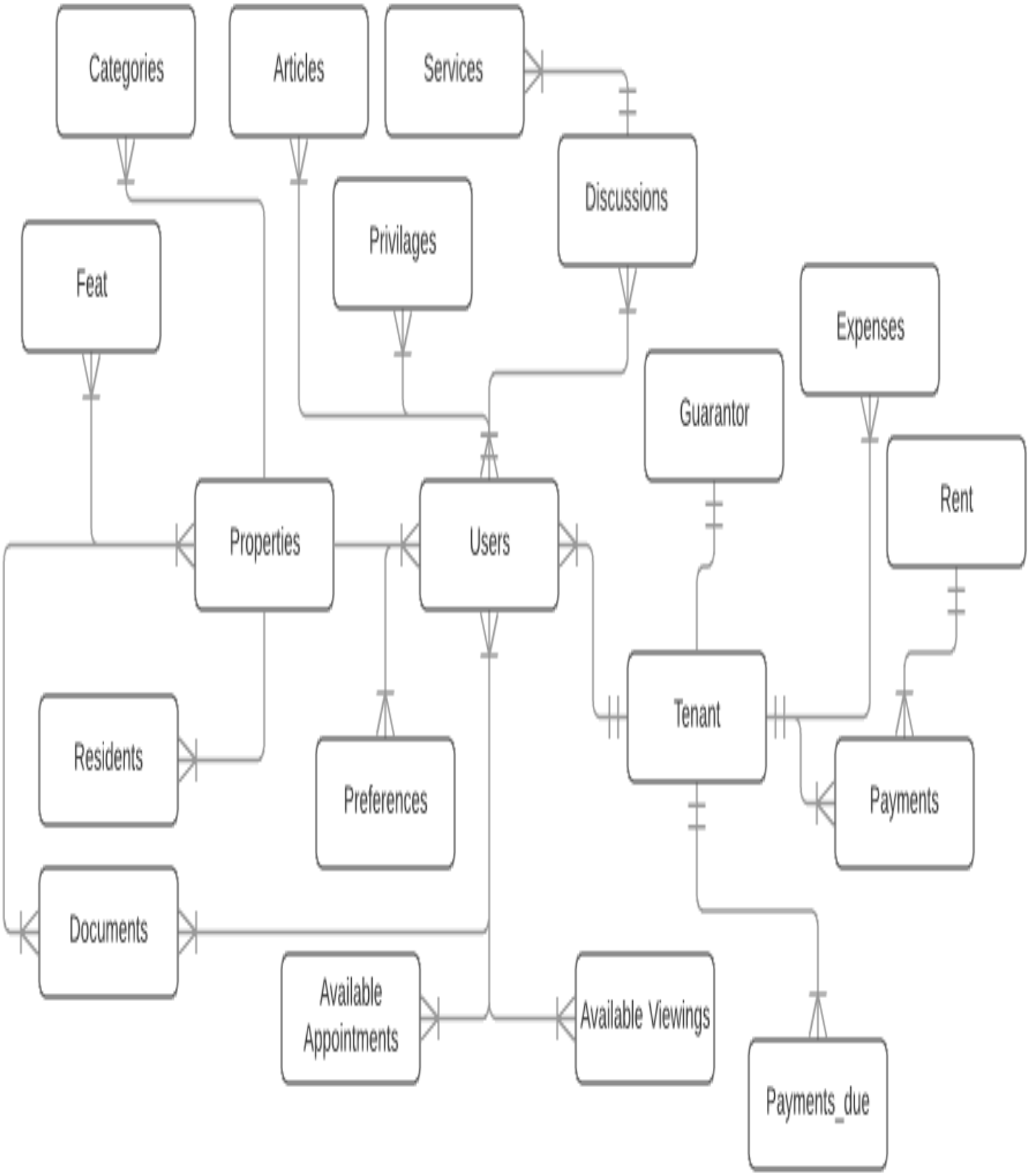
Admin_avail_appointments Table:

#	Name	Type	Null	Default	Extra
1	Id	Int(11)	No	*None*	AUTO_INCREMENT
2	Admin_id	Int(11)	No	*None*	
3	Date	Varchar(50)	No	*None*	
4	Time	Varchar(50)	No	*None*	
5	duration	Varchar(50)	No	*None*	

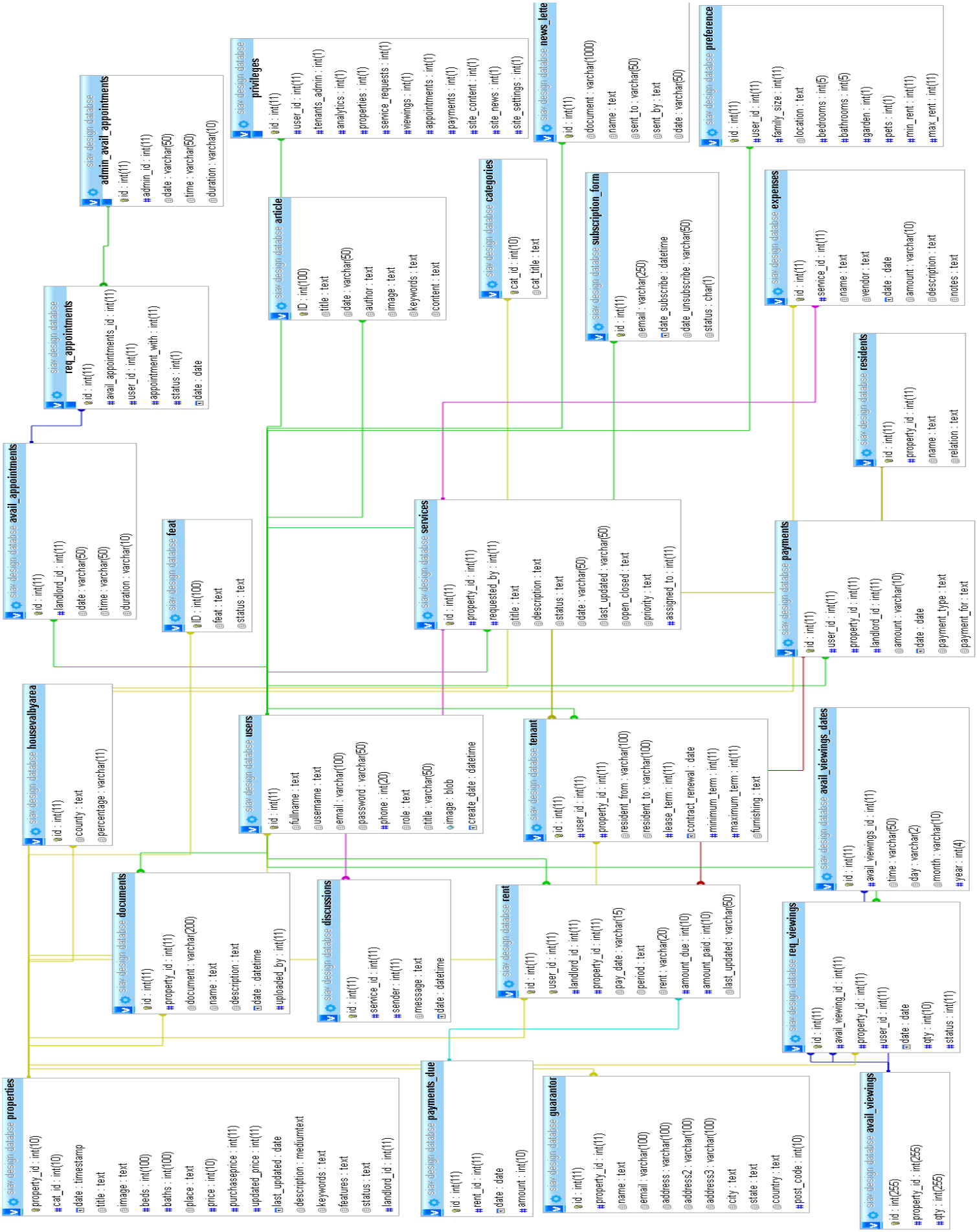
This table will store all the information about the admin available appointments dates, times and durations so people can book the viewings, linking to the admin_id on the from the users table.

Entity Relationship and Database linked Diagram

Entity Relationship Diagram



Database Linked Diagram



Database structure in standard notation

- **admin_avail_appointments** (**id**, admin_id, date, time, duration**)**
- **article (ID**, title, date, author, image, keywords, content**)**
- **avail_appointments (id, landlord_id,** date, time, duration**)**
- **avail_viewings (id, property_id,** qty**)**
- **avail_viewings_dates (id, avail_viewings_id,** time, day, month, year**)**
- **categories (cat_id,** cat_title**)**
- **discussions (id, service_id,** sender, message, date**)**
- **documents (id, property_id,** document, name, description, date, uploaded_by**)**
- **expenses (id, service_id,** name, vendor, date, amount, description, notes**)**
- **feat (ID,** feat, status**)**
- **guarantor (id, property_id,** name, email, address, address 2, address 3, city, state, country, post_code**)**
- **housevalbyarea (id,** county, percentage**)**
- **news_letter (id,** document, name, sent_to, sent_by, date**)**
- **payments (id, user_id, property_id, landlord_id,** amount, date, payment_type, payment_for**)**
- **payments_due (id, rent_id,** date, amount**)**
- **preferences (id, user_id,** family_size, location, bedrooms, bathrooms, garden, pets, min_rent, max_rent**)**
- **privilages (id, user_id,**tenants_admin, analytics, properties, service_requests, viewings, appointmnets, payments, site_content, site_news, site_settings**)**
- **properties (property_id, cat_id,** date, title, image, beds, baths, place, price, purchase_rice, updated_price, last_updated, description, keywords, features, status, **landlord_id)**
- **rent (id, user_id, landlord_id, property_id,** pay_date, period, rent, amount_due, amount_paid, last_updated**)**
- **req_appointments (id, avail_appointments_id, user_id,** status, date**)**
- **req_viewings (id, avail_viewiings_id, property_id, user_id,** date, qty, status**)**
- **residents (id, property_id,** name, relation**)**
- **services (id, property_id,** requested_by, title, description, status, date, last_updated, open_close, priority, assigned_to**)**
- **subscription_form (id,** email, date_susbcribed, date_unsusbcribed, status**)**
- **tenant (id, user_id, property_id,** resident_form, resident_to, lease_term, contract_renewal, minimum_term, maximum_term, furnishing**)**
- **users (id,** fullname, username, email, password, phone, role, title, image, create_date**)**

SQL Queries:

For the main concept of professional and good standard of programming, I will be using some complex queries to interact and fetch data from the database tables. The way of doing this would be to use linked SQL queries with the use of **JOIN, INNER JOIN, and LEFT JOIN** structure, instead of writing multiple queries, this would not only make the code clean but would also make the data entry more efficient.

An examples of Queries, I would be using for one of the core function of the website would be fetching and requesting property viewing dates. For this core function I would need to fetch data from multiple tables such as the avail_viewings (available viewings), req_viewings (request viewings) and avail_viewings_dates (available viewing dates). The queries I would use for this would be as follow:

For the **viewing_dates** function, I would have to link the avail_viewings and the avail_viewings_dates tables together to get the exact dates for each property viewing as the database tables have been simplified in 3NF, which means the data is more proficient and there is no duplication. So to do this will have to write a SQL query like:

```
$sql = "SELECT avail_viewings.id, avail_viewings.qty, avail_viewings.property_id,
            avail_viewings_dates.avail_viewings_id, avail_viewings_dates.time, avail_viewings_dates.day,
            avail_viewings_dates.month, avail_viewings_dates.year
        FROM avail_viewings
        LEFT JOIN avail_viewings_dates ON avail_viewings.id = avail_viewings_dates.avail_viewings_id
        WHERE avail_viewings.property_id = '$property_id'
            AND (SELECT COUNT(*) FROM req_viewings
                WHERE req_viewings.avail_viewing_id = avail_viewings.id
                AND req_viewings.status = '1') < avail_viewings.qty";
```

In this query I'm linking the table avail_viewings with the table avail_viewings_dates, and this is done by using left join to connect the tables and linking both tables where the **id** in the avail_viewings table matches the **avail_viewings_id** in the avail_viewings_dates table where the id is also equal to the property_id which will be fetched using the URL of the user browser, and also where the quantity of the allowed booking per slot is not exceeded by the allowed quantity set by the admin and this is check by counting the entries for the same property viewing slots requests and matching them with the status 1, if the allowed quantity is more than the set quantity the viewings details won't be shown to the users in the front-end.

For the **getRequestedViewings** function, I would have to link the req_viewings, properties and users table to get the viewing requests made by users for admins to review to approve them or reject them. So to do this I would have to do something like:

```
$sql = "SELECT req_viewings.date, req_viewings.qty, req_viewings.id, req_viewings.status, req_viewings.property_id,
            properties.title,
            users.fullname, users.email
        FROM req_viewings
        INNER JOIN properties ON properties.property_id = req_viewings.property_id
        INNER JOIN users ON users.id = req_viewings.user_id
        WHERE $option";
```

In this query I have extracted user requested entries from the req_viewings table where **$option** (requested id) matched the user and properties details in the properties and users table so when a requested viewing is accepted or rejected the users can be notified by email using data extracted from this query such as the **user's email.**