COMP9900 Project Report

Group AlphaGo

16 Nov, 2020

[Scrum Master]

Wanchao Huang	Front-end	z5192415	z5192415@ad.unsw.edu.au

[Group Member]

Raymond Lu	Front-end	z5277884	z5277884@ad.unsw.edu.au
Tao Xue	Back-end	z5219280	z5219280@ad.unsw.edu.au
Xiaohan Zhu	Back-end	z5187021	z5187021@ad.unsw.edu.au
Ziqi Liang	Front-end	z5202297	z5202297@ad.unsw.edu.au

Contents

1 Overview

1.1 Background

The Property Technology is changing the ways that auction firms cope with their business. A growing number of real estate is auctioned online through the third party services such as Openn, AnyWhereAuctions, and SoldOnline. Investment in property technology industry keeps increasing, and it is reported that by 2020, the investment reaches the level of $20 billion globally (Tan, 2017). However, on these platforms, properties are only sold within 90 days by agents, and there are no recommendation systems that can display a user properties in high similarity to the ones he/she has browsed.

For sellers, this change has substantially improved their experience and revenue. There is less stress on sellers as anonymity and privacy for sellers are guaranteed, buyer competition maximises the result, and auction dates have more flexibility. Moreover, instead of paying agents for administrating properties, sellers can monitor auctions easily whenever they have access to the Internet. On average, a property could be sold within 90 days via online auction once the owner engages the auctioneer (Lee & Adrian, 2017). Nevertheless, in full utilization of the Internet, there is space for shortening the auction period.

For customers, everything of an auction is transparent online as market comparisons among properties could be easily made. Besides, the time and space flexibility enables them to bid properties around the country at any time they are convenient, while, in the past, a potential customer lost his/her opportunity to bid if the auction location or time did not fit him/her. However, existing platforms make customers drown in uncategorised information and customers are starving for customised services that know what exactly they want. As when a platform starts to provide any unstandardized products to customers, other firms can no longer be competitive (Kotha, 1995), if a recommendation system which records customers' behaviours and preference and gives customers precise suggestions is available on an auction site, it is supposed to enable the platform to be outstanding in the market.

Those observations drive us to develop a property auction web portal, specifically for the real estate in Australia. The basic functionalities include search module, bidding module, property management module, notification module, and auction management module. Furthermore, a recommendation system is our novelty that attracts users to use with the merit that they do not need to search properties under a certain criteria repeatedly. Saving customers' searching time also has the potential to boost up the auction speed. Highlight as our design on synchronization is, compared to contemporary online auction systems, our exclusive introduction of WebSocket guarantees all users can get the latest auction information without refreshing the auction page.

1.2 Software Architecture

As the software architecture Figure 1.2.1 shown below, the project applies the common three-tier architecture, namely the presentation tier, application tier, and data tier. A bottom-up approach is applied to demonstrate the design in the following paragraphs.

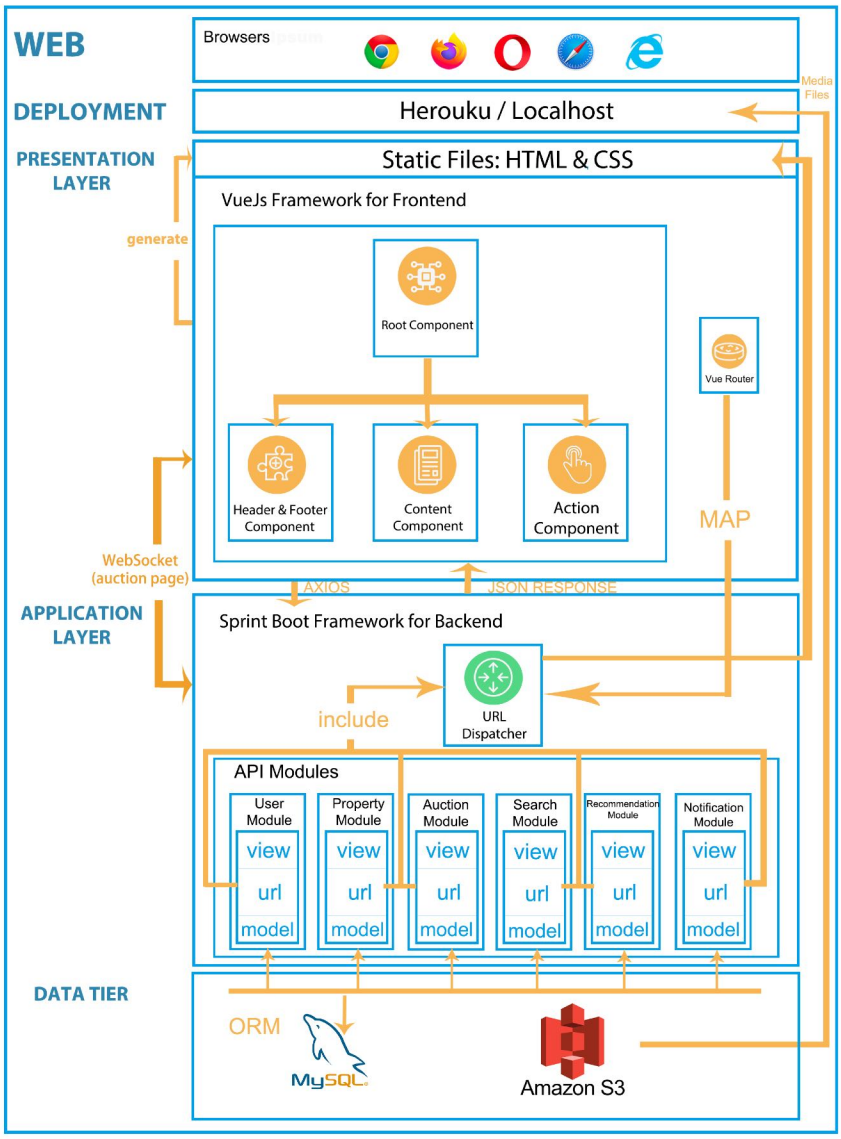


Figure 1.2.1 Software Architecture

1.2.1 Data Tier

Initially, the data tier is to be discussed. In our design, considering the comparatively long reading and writing time in the interaction with database when it comes to large data such as images, the cloud computing platform is introduced to boost up the per-

formance. We choose Amazon S3 as the cloud storage service provider, and every time a user uploads images to the back-end, they are saved to the cloud and the back-end database only stores the image storage URLs. By doing this, every time web pages from different corners around the world can load those static data in a short time since Amazon deploys server at many cities, which means data could be transferred from the closest server to the query point instead of from the specific database server at somewhere. Moreover, wide bandwidth of Amazon guarantees fast connectivity, and Amazon S3 has the merits of data security and scalability. However, in terms of logic schema design, the traditional relational database is still required to represent the relations between instances. We choose MySQL in this project to store text information. Based on the choice of MySQL, the MyBatis framework, which couples objects with stored procedures or SQL statements using an XML descriptor or annotations, is used to simplify programs and improve development efficiency. Furthermore, Redis and RabbitMQ are introduced, with the former to relieve the reading and writing burden to database as it stores data in cache, and the latter to improve asynchronousity and concurrency. Besides, local storage is utilized to store user tokens with which users do not need to type in the same credential details repeatedly. *Vuex* under the Vue framework is utilized for local cookies which saves login information. An image illustrating the data tier is given below.

Figure 1.2.1.1 Data Access

1.2.2 Application Tier

When it comes to the application tier, AlphaGo Auctions's back-end is constructed upon the Java Sprint Boot framework. For decoupling, a MVC three-layer architecture is applied as following:

- Model Layer: Data persistence and process requests.
  - Service Layer: Deal with specific business.
  - Data Access Object: Communicate with database.
  - Plain Java Objects
- Controller Layer: Receives HTTP request based on the url from the Front-end and returns results from the Model layer.

- View Layer: Data representation. In our architecture, it is replace by the front-end system

However, slightly different from the traditional MVC structure, we choose a Front-Back-Separation structure which means the Front-end server (Section 1.2.3) replaces the the view layer. With this, the back-end side only needs to focus on function logic and implementation.

Once the back-end receives the request from the front-end(*view layer*), it dispatches the requests to one of controller according to the request URL. Then each controller can call corresponding services to handle requests.

1.2.3 Auxiliary Frameworks and Techniques

In our system's application tier, we use frameworks and techniques to improve the availability.

- Shiro

  Shiro is an open source Authentication and Authorization framework. With the help of Shiro, our system has a robust user authentication procedure.

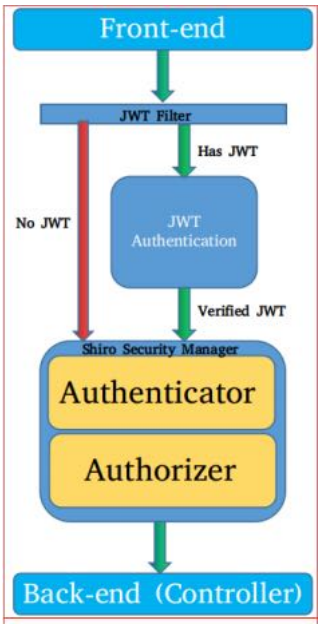


Figure 1.2.3.1 Authentication Procedure

- Redis

  Redis is very popular NoSQL database and in our system the Redis has two functionalities:

  - Database Cache: To avoid overload on our database, we use Redis to play the role of database cache. With that, the cost of reading operations can be alleviated.
  - Timer: As our system has very strict demands in real-time synchronization, Redis is very helpful because of its key expiration function.

- RabbitMQ: RabbitMQ is a common use open source message queue, which can improve the system's concurrency.

1.2.4 Presentation Tier

As for the presentation tier, Vue.js framework is used for the front-end design for the sake of easy integration with other external libraries, better combination of CSS and JavaScript, and simpler coding grammar. Each web page has its own *.vue* file and some external components that are introduced to realize different functionalities in it. Vue router maps the routes and lets the back-end know where to find the corresponding view to render the contents. Communications between the front-end and the back-end are implemented via Axios, a Promised based HTTP client for JavaScript. In the majority of cases the front-end makes requests like GET and POST to the back-end, and the back-end responds with a JSON data.

2 Data Model

2.1 Data Preparation

Before testing the functionalities of this project, some property data is required to be stored in the database to imitate the real scenario. To get some real property information, we sampled 72 real estate from *domain.com*, save those real estate's photos, and registered them on our platform. An sample from *domain.com* is given below.

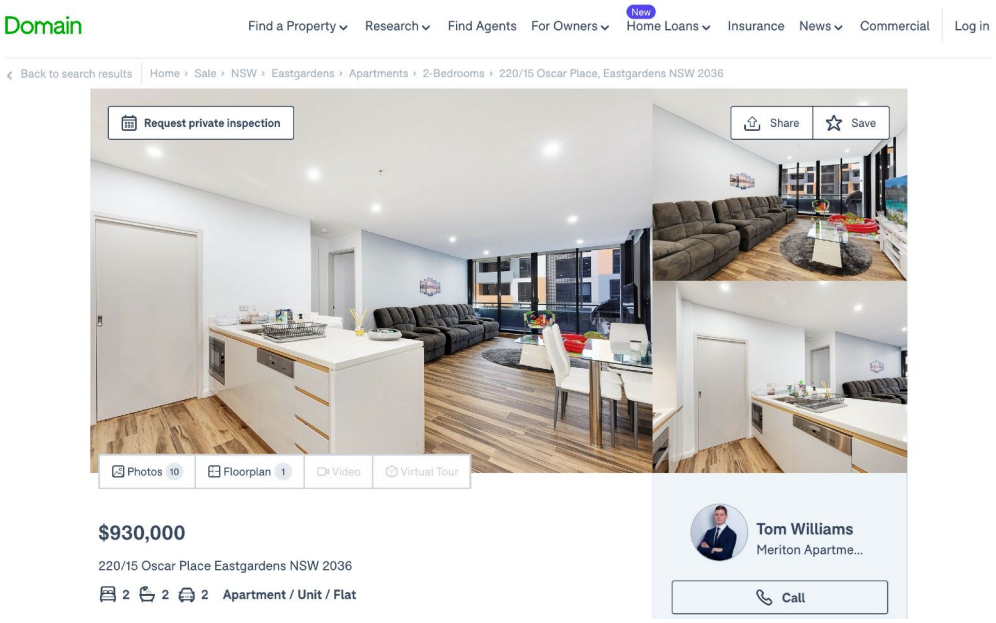


Figure 2.1.1 A Sample from Domain.com

Moreover, we created 10 user accounts, and each account has its own identity. There are four identities in total: visitor, registered auction bidder, seller, and a mix of the second and the third. Different accounts with different identities should have different accesses to properties, and this was checked in the debug phase.

2.2 Database Architecture

As mentioned above, MySQL is used to store text data, and Amazon S3 is used for image data. After uploading images to Amazon S3, URLs for images will be generated and they are saved in the MySQL database. The local schema is represented in the MySQL, and the structure is depicted in the image below.

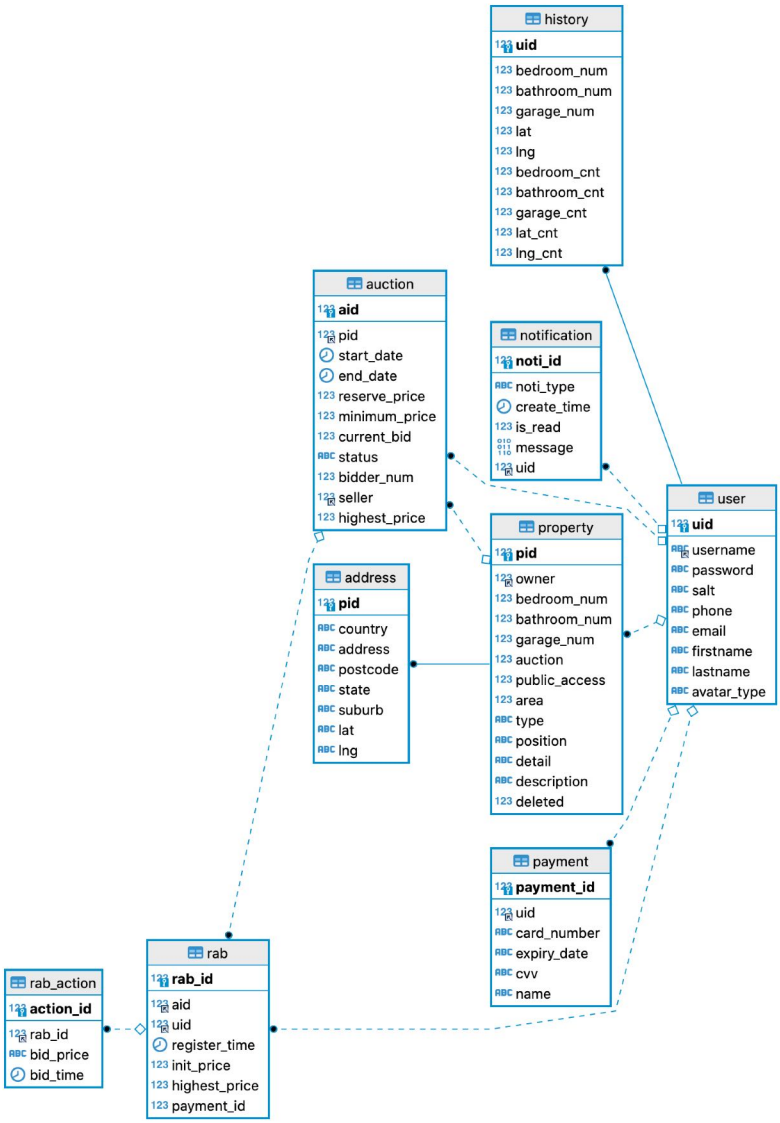


Figure 2.2.1 MySQL Database Schema

2.2.1 Table USER

The **USER** table stores the user information, with *uid* is the primary key for users and the *password* saved in the database is encrypted. The *avatar_type* represents the file format of avatar, such as jpg or png. To save the storage space of the MySQL database, some user information is stored locally on the user's machine. The link to the avatar image stored in the Amazon S3 is saved in Cookies. In case of repeat login procedures, we save the JWT tokens locally on user's machine so that after the first login, the browser can load the user's token from Cookies for identification.

2.2.2 Table HISTORY

The **HISTORY** table records the behaviours of users. By recording the details of each searching, filtering and auction registration every time, the recommendation system has an approach to know which kinds of properties are favored by a certain user, based on this table.

For the attributes *bedroom_num, bathroom_num, garage_num, lat, lng*, they will increase if a user:

1. Participates in a property's auction as a registered auction bidder. These values will augment by the corresponding attributes of the auctioned property.
2. Applies corresponding filters. For example, if a user applies a filter condition of *bedroom number =2* in a search, the *bedroom_num* will increase by 2.
3. Search properties with a suburb name. After such a search, we get the latitude and longitude values of the searched suburb, and add these two values to the *lat, lng* attributes respectively.

For attributes *bedroom_cnt, bathroom_cnt, garage_cnt*, they will increase by one if the user behaviour one or two above happens, while, for attributes *lat_cnt, lng_cnt*, they will increase by one for conducting user behaviour one and three.

2.2.3 Table NOTIFICATION

The **NOTIFICATION** table records the historical notifications to a certain user. The *is_read* plays a reminder role in the project that if its value is zero, a red dot will pop up above the user's avatar to reveal that a new message is not read yet. The *message* attribute is a BLOB object so that long text information could be saved in a comparatively small memory consumption.

2.2.4 Table PAYMENT

The **PAYMENT** table records the details of debit and credit cards owned by different users. With card information saved, users do not need to input card credentials repeatedly.

2.2.5 Table PROPERTY and ADDRESS

Another significant table **PROPERTY**, with *pid* as its primary key, saves the intrinsic attributes of real estate such the number of bedroom. In order to have a well structured property address information saved in the database, an auxiliary table **ADDRESS** is designed for well maintenance.

2.2.6 Table AUCTION

Combining the information of the table **PROPERTY** and **USER**, the **AUCTION** table plays the most important role in the whole system as it records the auction details. For clear specification, *reserve_price* represents the price threshold that the seller is willing to accept in the auction; *minimum_price* represents the lowest price from which the auction should start; *current_bid* is the ID of the user who has made the highest bid so far; *status* represents the current status of auction, with 'A' meaning under auction, 'S' meaning successfully sold, 'R' meaning registered but not commenced yet, and 'F' meaning auction passed in; *bidder_num* indicates the total number of participated bidders; *seller* represents the seller's user ID; *highest_price* represents the current highest bid in the property's auction.

2.2.7 Table RAB and RAB_ACTION

Since the bidding history is required to be displayed on the auction page and be sent to the seller if the property eventually passes in, the actions of each registered auction bidder are needed to be recorded in a fine granularity. The **RAB_ACTION** table is introduced to record all actions made by a single user in an individual auction, while, the **RAB** table, which has a foreign key pointing to the former, keeps the general bidding information of a user in a certain auction.

3 Functionalities

3.1 User Authentication

For most pages like Profile page and Property Management page, users should log in to get the permission to view them. Thus we need to check the login status after the route switches. If the user has not logged in but gets directed to a specific page through some tricky ways (such as directly modifying the route), we will pop up an error reminder and push the user to login Interface (as shown below).

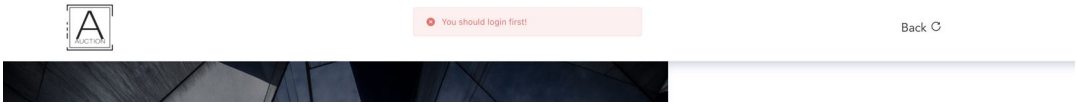


Figure 3.1.1 Error Example

Whenever a user logs in, his/her avatar will be displayed at the top right corner. When the mouse hovers over the avatar, a drop-down menu will appear. This menu is the main navigation tool of our website. Users can click on the options in the drop-down menu to get directed to the target page.

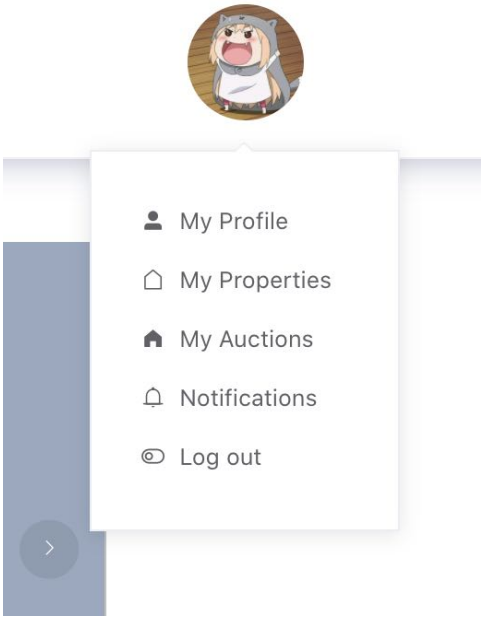


Figure 3.1.1 Drop-down Menu at Avatar

3.1.1 Register

A new user can register a new account by simply clicking the button at the top right corner on some pages.

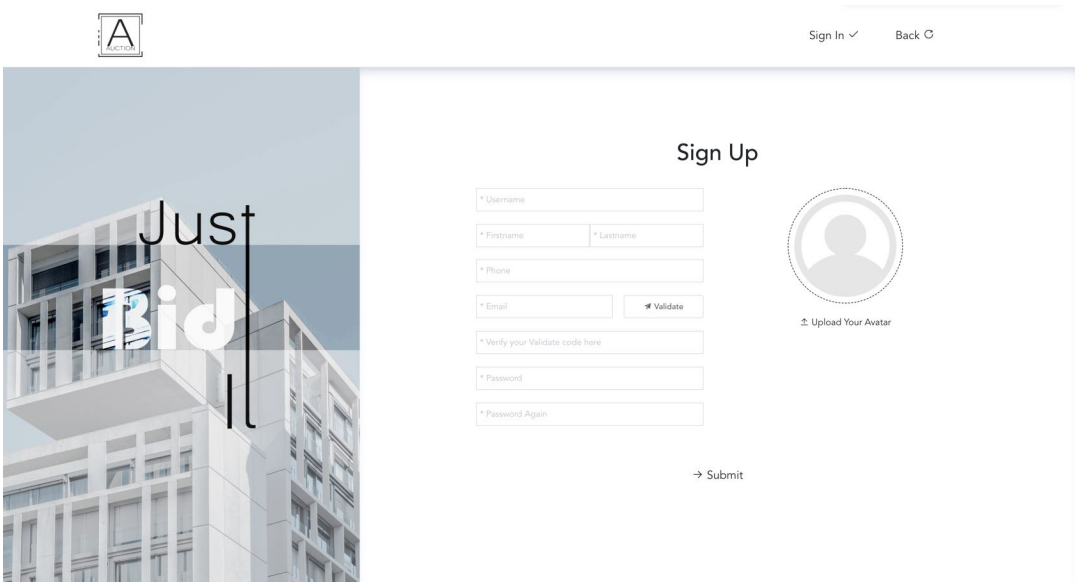


Figure 3.1.1.1 Register Page

In the new account application procedure, a user name which only consists of numbers and characters is required first. For easy system maintenance, the user name is unchangeable. Then, first name, last name, email address, phone number, password and confirmed password are needed. Special validation rules are applied to some of them. Email address is checked whether it is in the format of a normal address, and the phone number is verified to belong to Australia. Password must include at least one numeric digit, one uppercase letter, and one uppercase letter, or the validation of it fails. Finally, the confirmed password will be checked of its coherence to the initially set password.

Before submitting the information, the user needs to verify his/her email address by clicking the "Validate" button. The back-end will send a validation code to the to-be-validated email address. If the correct validation code is submitted back to the back-end within the limited time and the email address has not been registered before, the validation procedure passes. Afterwards, with the remaining information submitted to the back-end server, a new account is created successfully.

If the user does not upload an avatar, we will automatically generate a default avatar based on the first letter of his/her first name after he/she submits all the information.

Figure 3.1.1.2 Default Avatar

3.1.2 Login

If the user has already registered an account on our platform, he/she can choose to login by username or email address on login page. Once the information is validated by the back-end, the login procedure is successful. We use *localStorage* to store the user's login information. If the user closes the browser after logging in, there will be no need to log in when opening the window again.

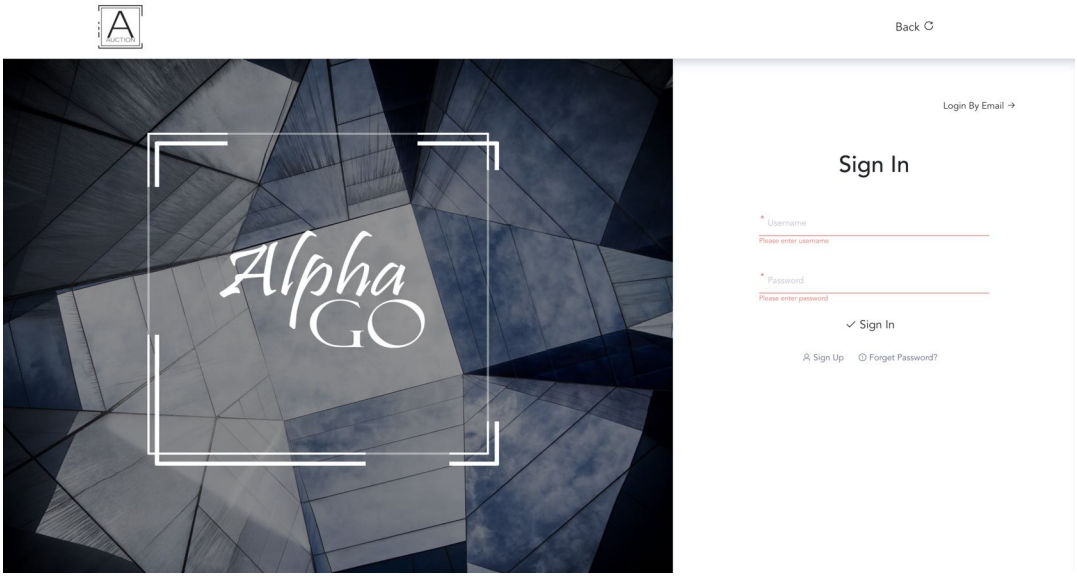


Figure 3.1.2.1 Login Page

3.1.3 Reset Password

If a user forgets his/her password, he/she can click the "forget password" button and go to the resetting page. Before submitting the request, the system needs to verify his/her identification first. Similar to the registration process, he/she needs to attain a validation code by clicking the "Validate" button and to type in the code. If the validation code is correct, the system will proceed with the reset password request.

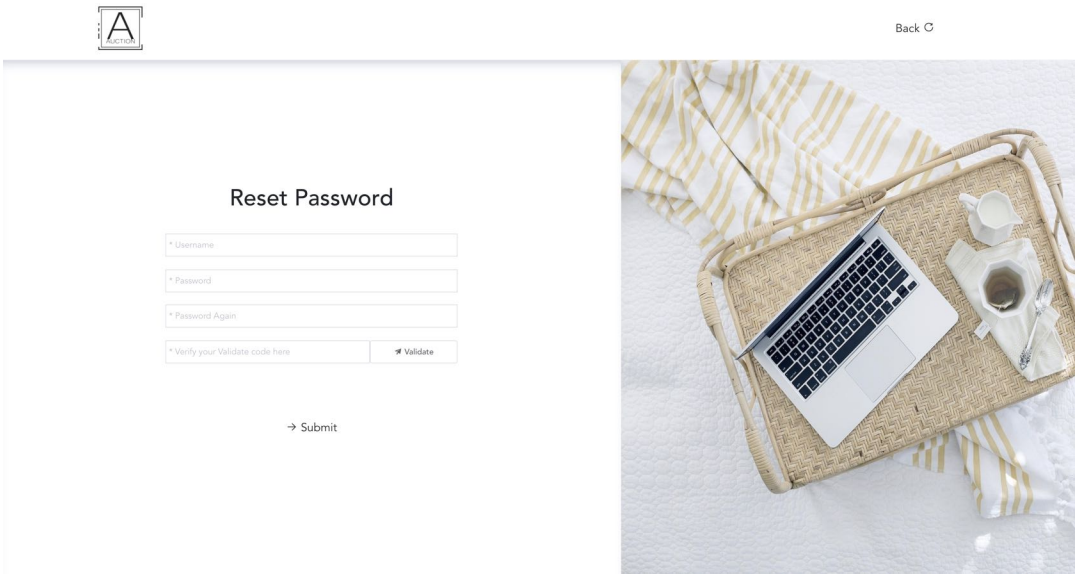


Figure 3.1.3.1 Reset Password Page

3.1.4 Profile

Once a user has already logged in our website, his/her avatar will be shown at the top right corner of the header navigation bar. By hovering the mouse on the avatar, there will be a drop-down menu, the first row is "My Profile", which will direct the user to their profile pages.

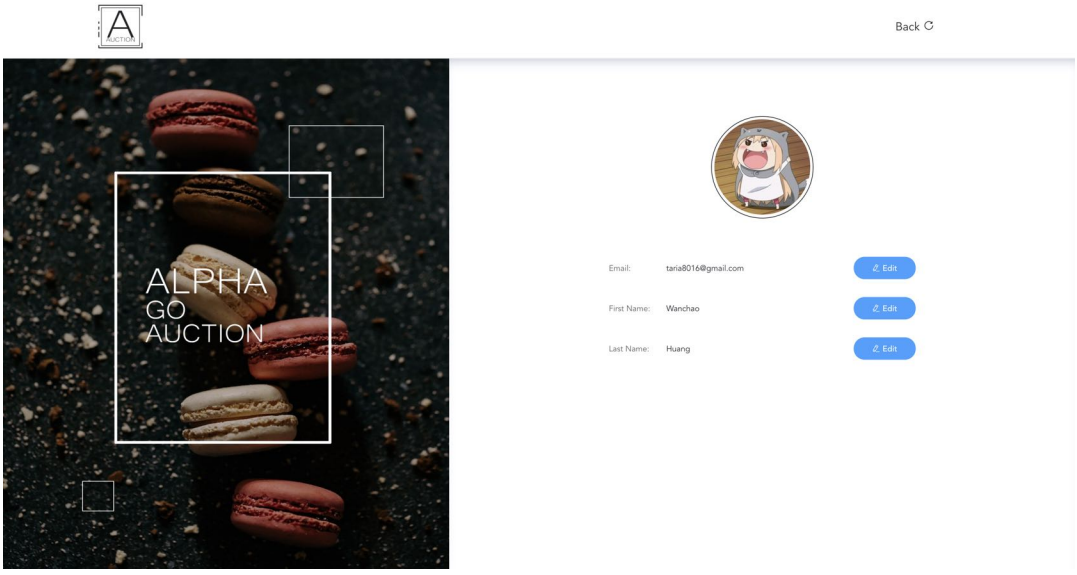


Figure 3.1.4.1 Profile Page

On the Profile page, users can change their information except for the username. If

they intend to change the email address, the same validation process as the password resetting will be also needed.

In this section, we also provide the avatar change. When a user uploads a new avatar, there will show up two buttons: one is for canceling the upload while another one is used for confirmation of the change action.

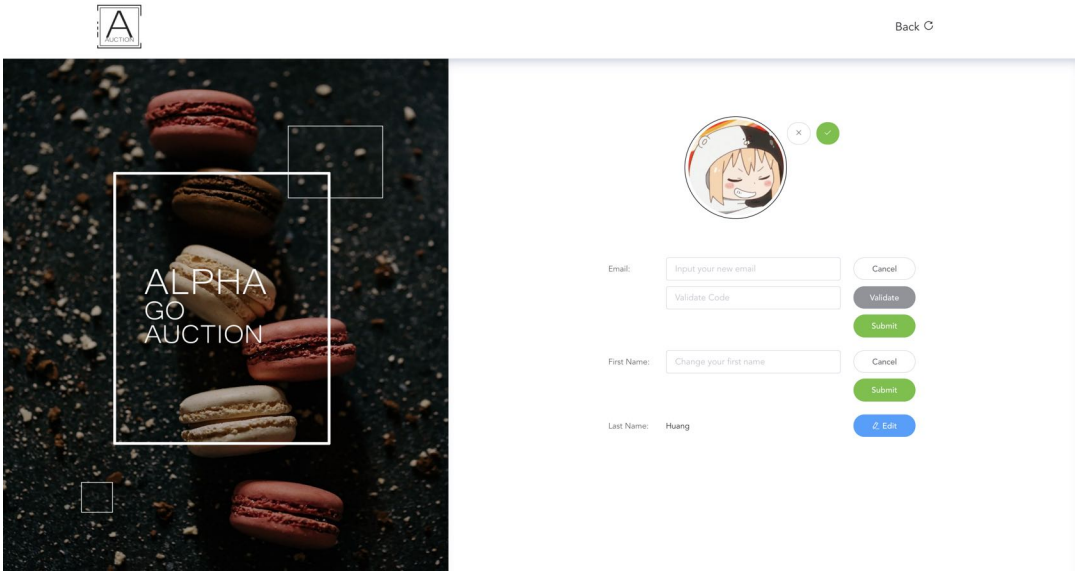


Figure 3.1.4.2 Profile Page Interaction

3.2 Search

3.2.1 Home Page

A search bar is designed at the center of the home page. A user can search properties only by suburb names or postcodes. The suburb name inputted by the user will be validated by Google Maps API first to guarantee such suburb exists, and the postcode will be checked whether it is a four digit number. Once the search condition is valid, the user will be redirected to the search page.

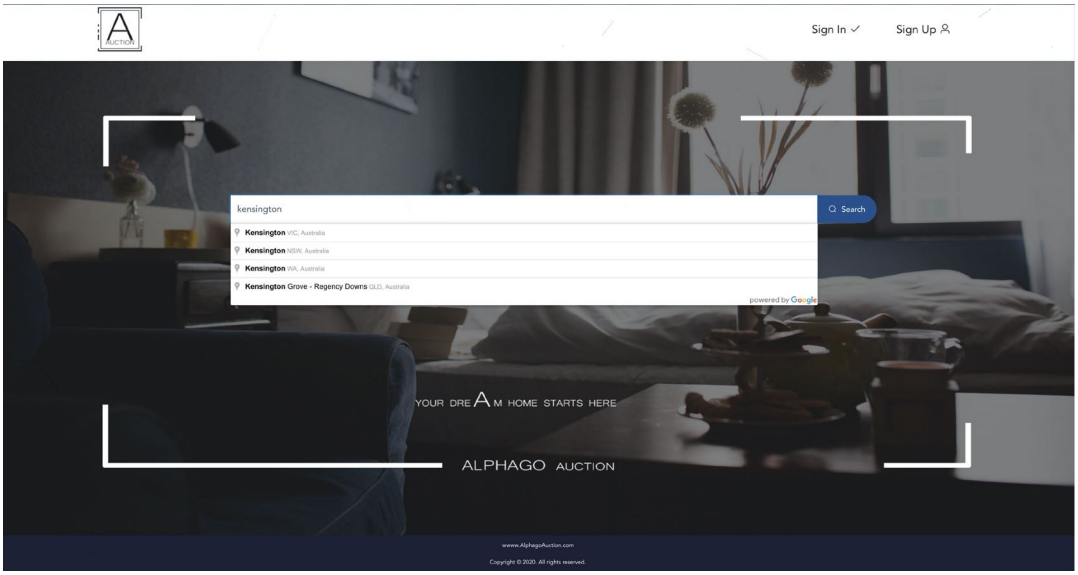


Figure 3.2.1.1 Search Suburb on Home Page

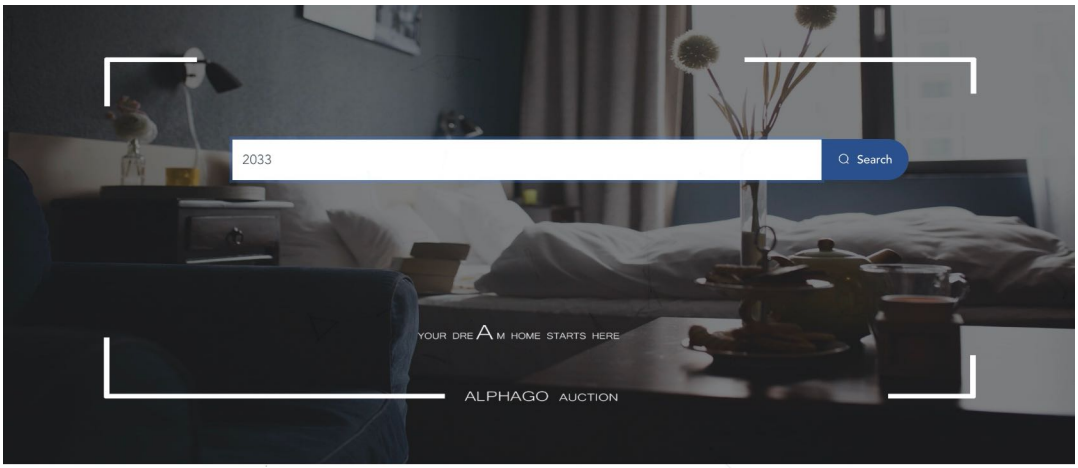


Figure 3.2.1.2 Search Postcode on Home Page

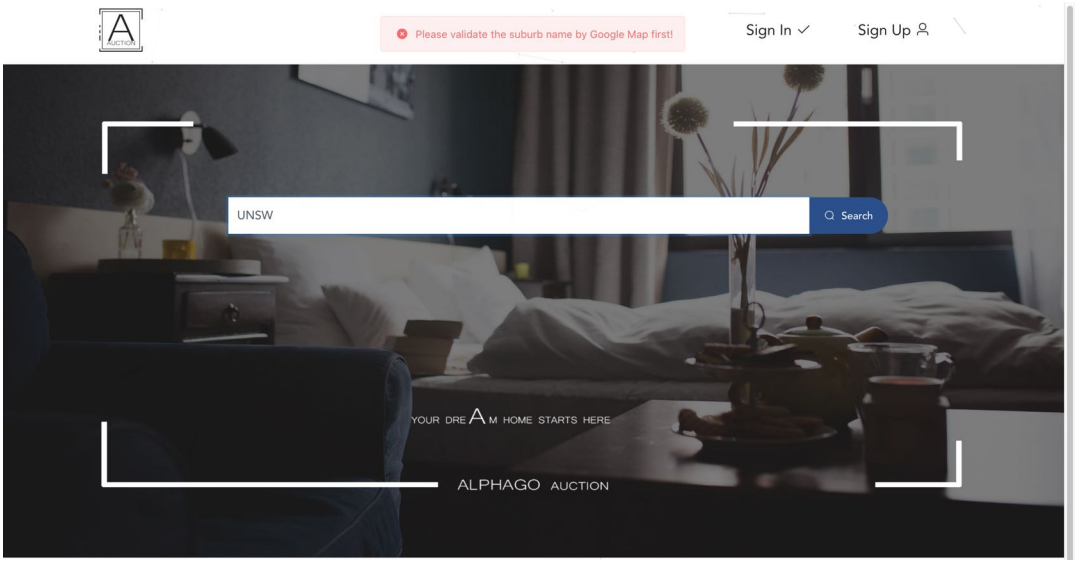


Figure 3.2.1.3 Invalid Suburb Error

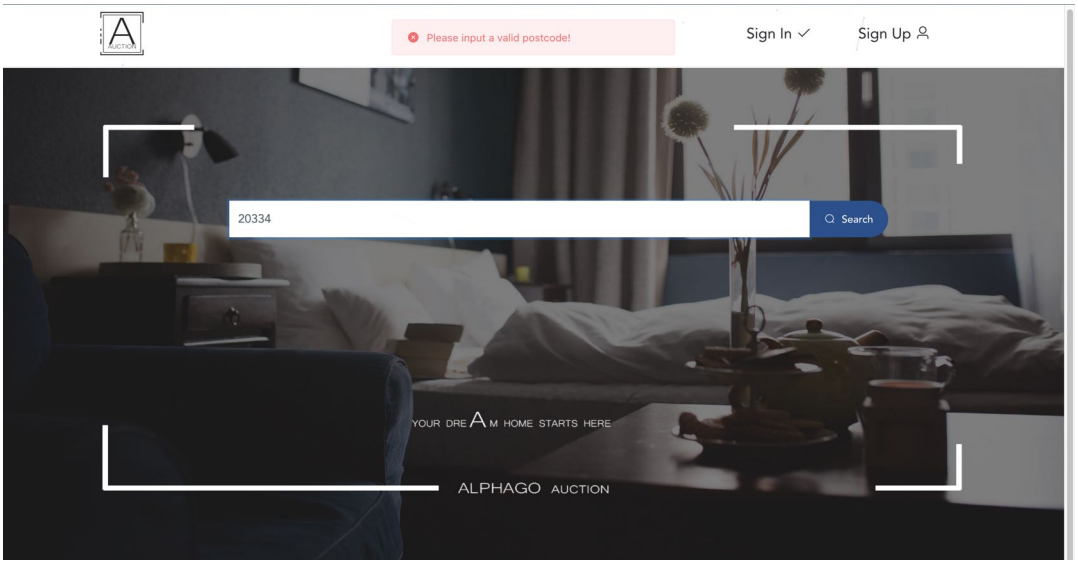


Figure 3.2.1.4 Invalid Postcode Error

3.2.2 Search Page

When it comes to the search page, the search bar functionality remains the same as the one on the home page. And now user can conduct finer search by applying filters to the search condition.

3.2.2.1 Filters

The provided filters are:

- Start Date: A date picker which is for choosing auctions that start after.
- End Date: A date picker which is for choosing the auctions that end before. We have applied a rule to it that this data must be later than the Start Date filter.
- Bedrooms: A drop-down menu for choosing the exact number of bedrooms the user wants the property has.
- Bathrooms: A drop-down menu for choosing the exact number of bathrooms the user wants the property has.
- Garages: A drop-down menu for choosing the exact number of garages the user wants the property has.
- Type: A drop-down menu for choosing the type of property the user wants the property is.
- Price: Two numerical inputs for the minimum price and the maximum price. A rule that the maximum price must be greater than the minimum price is applied. The valid value range of the minimum price is from $0 to the maximum price, while the one of the maximum price is from the minimum price to $100,000,000 AUD.
- Area: Two numerical inputs for the minimum area and the maximum area. A rule that the maximum area must be greater than the minimum area is applied. The valid value range of the minimum area is from 0 to the maximum area, while the one of the maximum area is from the minimum area to 100,000 m^2.
- Sort: A drop-down menu for choosing the sorting order in which the properties are displayed. Properties could be displayed in the low-to-high price order, the high-to-low price order, or the latest-to-oldest auction registered order.
- Clear Filter: This button is for resetting all filter values.

As the line of words below the filter hints, the filter conditions will be applied only if a user clicks the "Search" button. By doing this, the suffix of the front-end URL changes to the latest filter condition, and the front-end sends a POST request to the back-end to get the new search results. Since the filter conditions are all enumerated in the URL, if a user switches back to the search page from a certain auction page, the search results would remain the same.

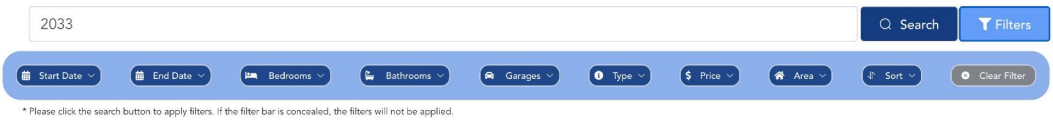


Figure 3.2.2.1.1 UI of Filter

3.2.2.2 Property List

Each property returned by the back-end is displayed in a card, and there are two different themes for cards representing the auction in process status (green) and incoming auction status (red).

1. Auction In Process

   The card consists of a thumbnail of property image, property address, property's basic information, auction end time and the number of attended bidders. Also, the current highest bid price will be displayed in a label in green color.

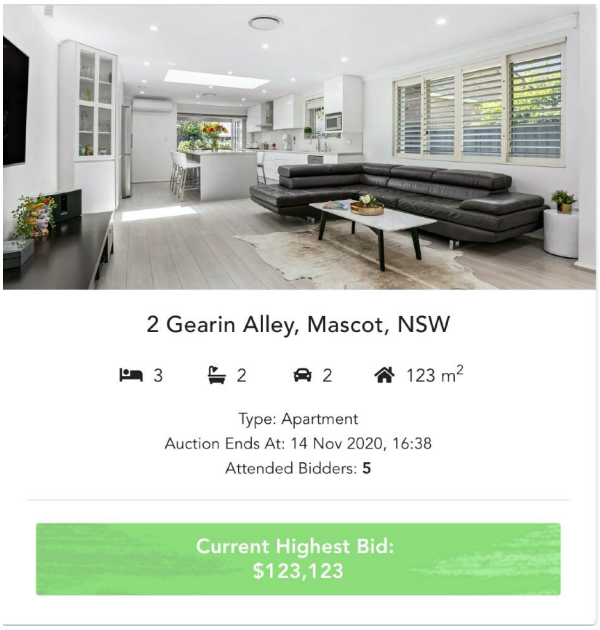


Figure 3.2.2.2.1 Auction In Process Example

2. Incoming Auction

   The card consists of a thumbnail of property image, property address, property's basic information, auction start time and the number of attended bidders. Also, the guided auction price will be displayed in a label in red color.

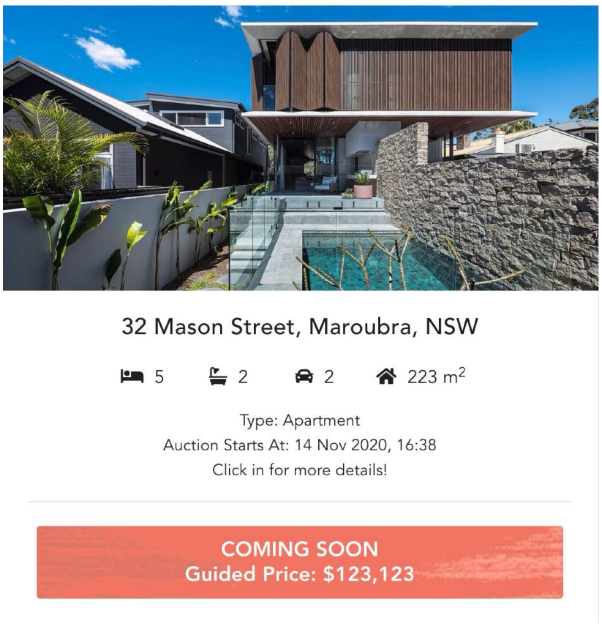


Figure 3.2.2.2.2 Incoming Auction Example

3.3 Property

3.3.1 Property Management

After logging in our website. Property Management page can be found by hovering on the avatar, and clicking the second option "My Properties".

3.3.1.1 Empty Property

If a user has not registered any properties yet, when he/she visits the property management pages, an empty alert will be triggered on the top of the window.

Figure 3.3.1.1.1 Empty Property

3.3.1.2 Management Menu

The property management page is mainly used to display the registered properties. The page layout is divided into two parts. The left one is a list of existing properties. At the top of the page on the left hand side, the user can add his property by clicking "+New" button and get redirected to the property registration page. By clicking on a certain property on the left, the detailed information of the property will be displayed on the right. The properties on the left are divided into three types according to the auction status: properties that have not been registered for any auctions, properties that have

been registered for auctions but the auctions have not started yet, and properties that have been registered for auctions and the auctions are in progress.

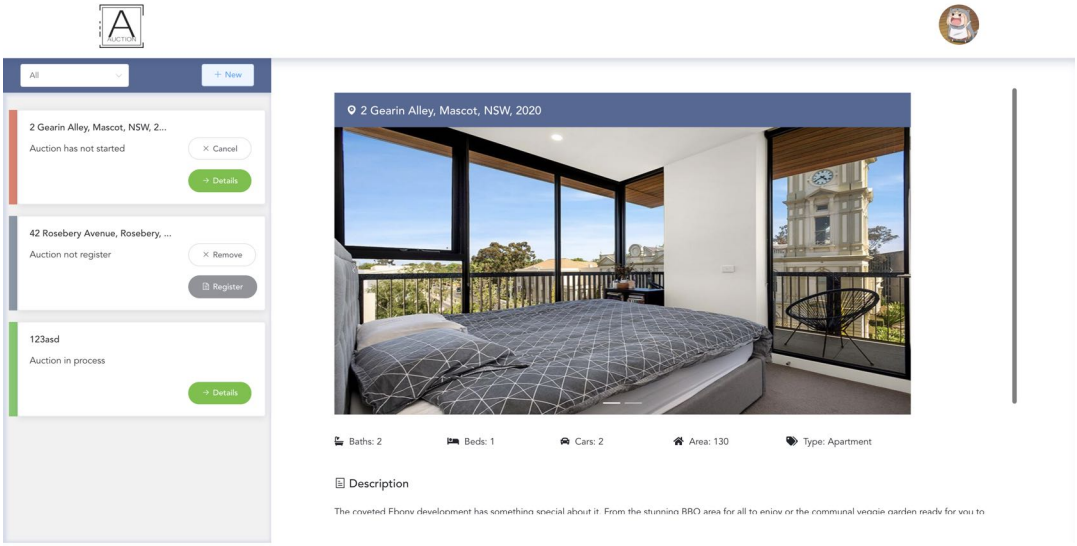


Figure 3.3.1.2.1 Property Management Page 1

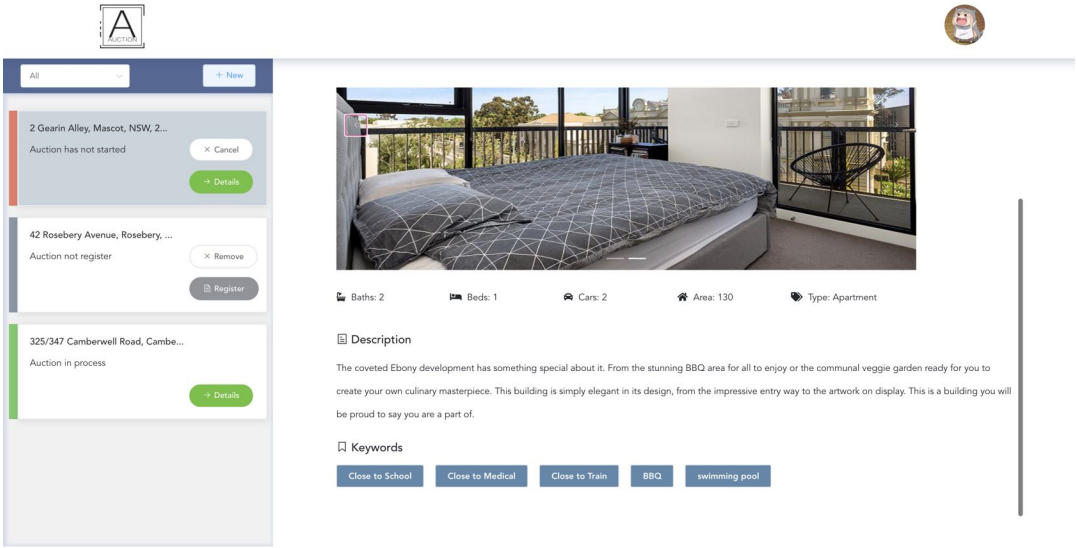


Figure 3.3.1.2.2 Property Management Page 2

In the management system ,there are three different kinds of property:

- Property without any auction registration: This kind of property is labelled with a grey badge on the left border, and there are two buttons on the right hand side

of the property menu card: "Remove" button for removing the indicated property, and "Register" button for redirecting to the auction registration page.

- Property that has been registered but has not started yet. This kind of property is labelled with a red badge on the left border, and there are two buttons on the right hand side of the property menu card: "Cancel" button for cancelling the current auction, and "Details" button for redirecting to the property auction page.
- Property that has commenced an auction. This kind of property is labelled with a green badge on the left border, and once the auction has started, the auction could not be cancelled by the seller directly considering it is not fair for a registered auction bidder who has made the current highest price. If a seller wants to cancel an on-going auction, he/she needs to contact the AlphgGo staffs. There is a button on the right hand side of the menu card: "Details" button which performs the same functionality as above.

3.3.1.3 Auction Registration

If a user clicks the "Register" button on a property's menu card, there will pop up a registration form to register an auction. The start date, end date, reserved price and starting price are needed to fill in, and the starting price should smaller than the reserved price. Besides, payment detail is needed in the registration. A user can choose a recorded bank card or add a new bank card information. Soon after the transaction is successful, money will be transferred to the registered bank card.

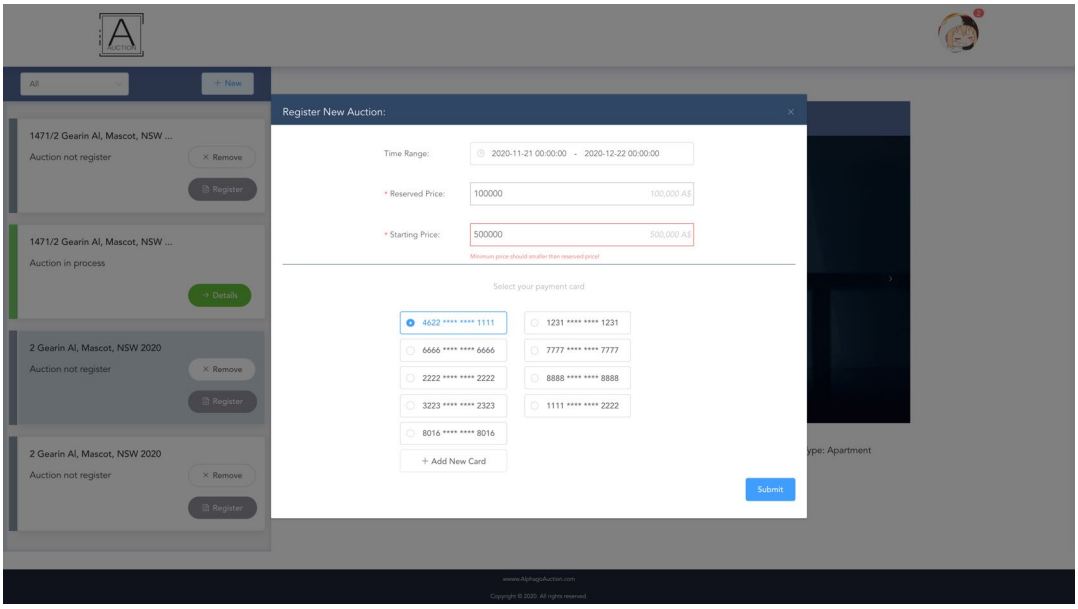


Figure 3.3.1.3.1 Auction Registration Form - Choose An Existing Card

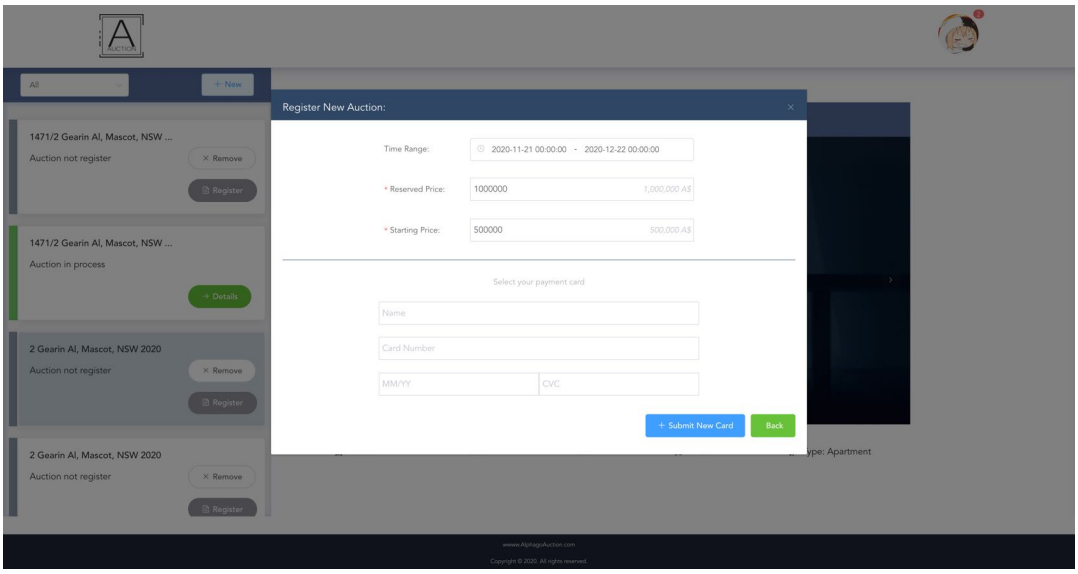


Figure 3.3.1.3.2 Auction Registration Form - Create A New Card

3.3.2 Property Registration

A logged-in user can access his/her properties by clicking the drop-down menu option "My Properties". To register a new property, the owner can click the "+New" button on the top left and go to the Property Registration page. On the Property Registration page, the user needs to provide all necessary information of his/her property. There are five steps in the registration process, and from the second step, each step is unlocked until the previous one is validated. Once a step is finished, the corresponding icon on the status bar will turn to green color from grey color. All registered properties can be reviewed on the property management pages.

3.3.2.1 Basic Information

The first session is address validation. We introduce the google auto-complete API to confirm the existence of a property. Once a user clicks the address shown in the drop-down menu, the input bar will be filled in with the chosen option, and the form below will be automatically filled too.

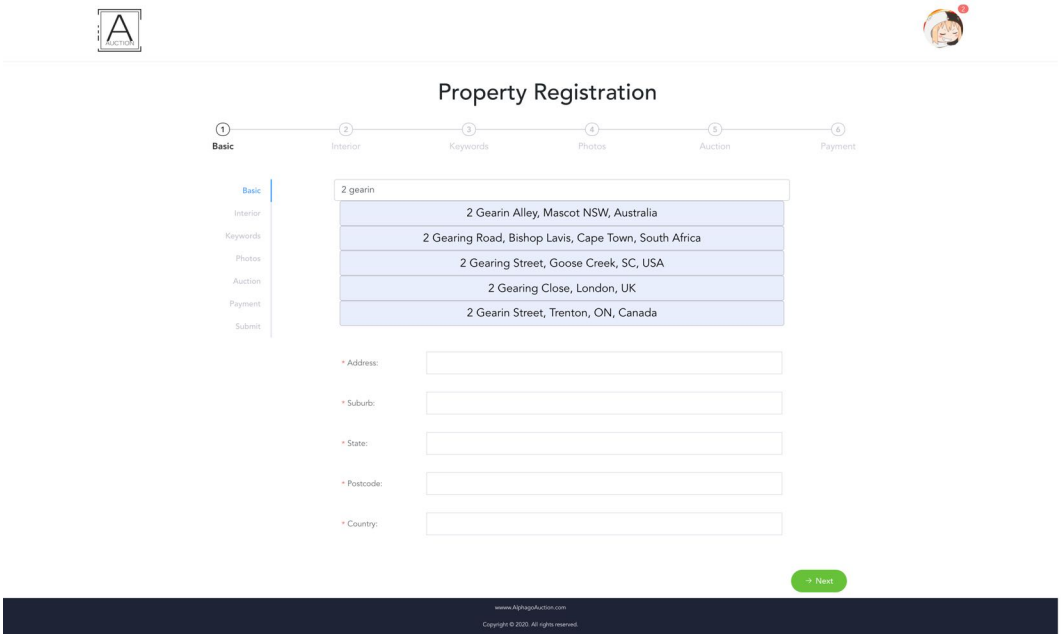


Figure 3.3.2.1.1 Property Registration Section 1

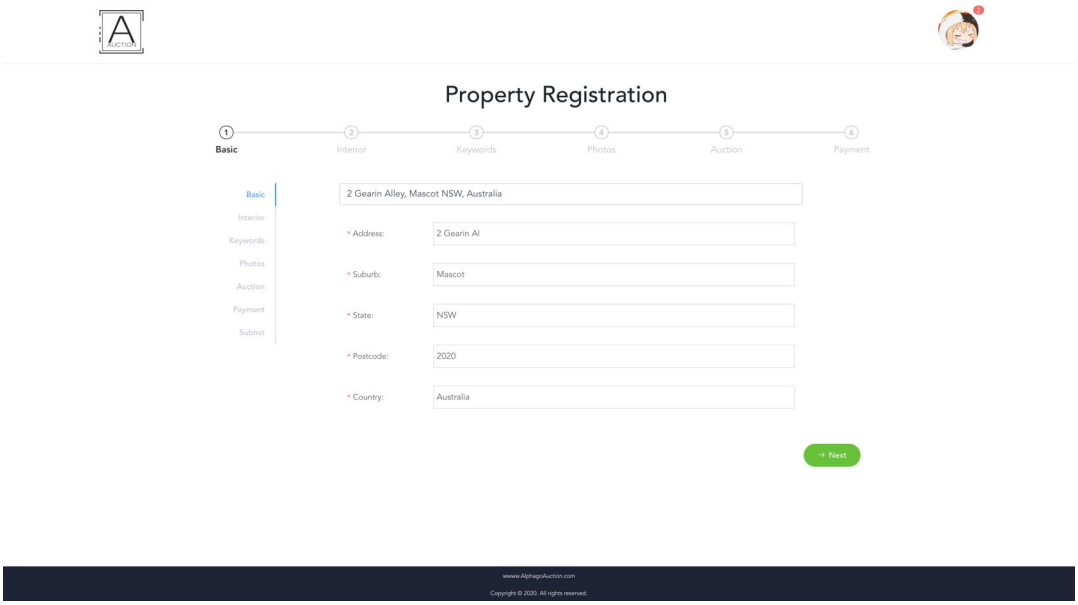


Figure 3.3.2.1.2 Property Registration Section 1 Auto Complete Function

3.3.2.2 Interior Information

The second part is to input interior information like property type, area, and room numbers. The type is limited in the following options: *House, Apartment, Studio, House and Commercial.* Area is supposed to be a numeric integer and no more than 5

digits. Same as the area, room number should also be a number and no more than 2 digits.

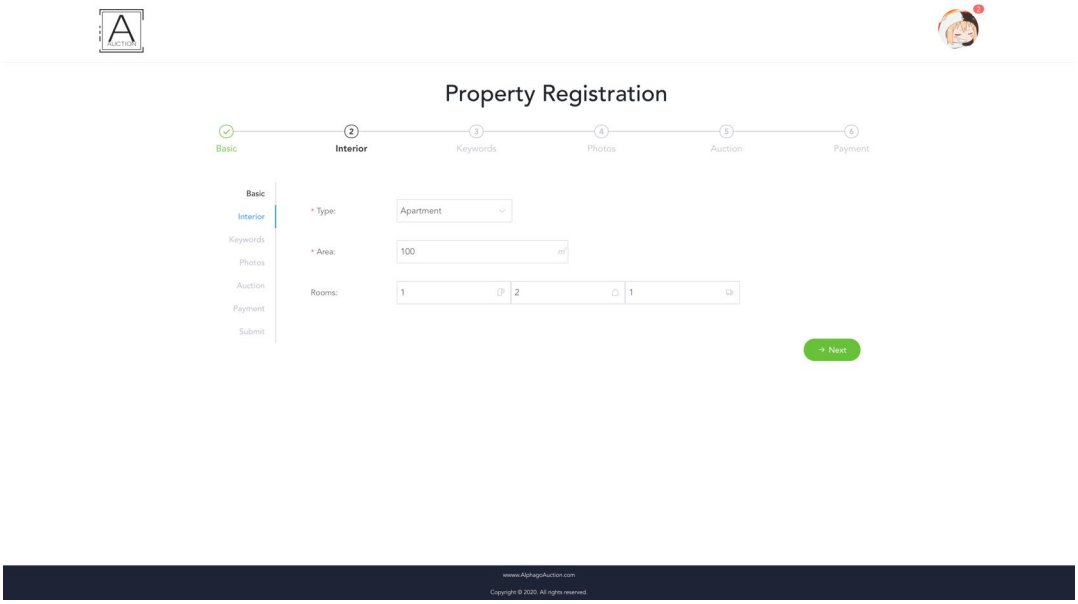


Figure 3.3.2.2.1 Property Registration Section 2

3.3.2.3 Keywords Information

The third part is to select the keywords and to write a description of your property. The description is limited in 1000 words.

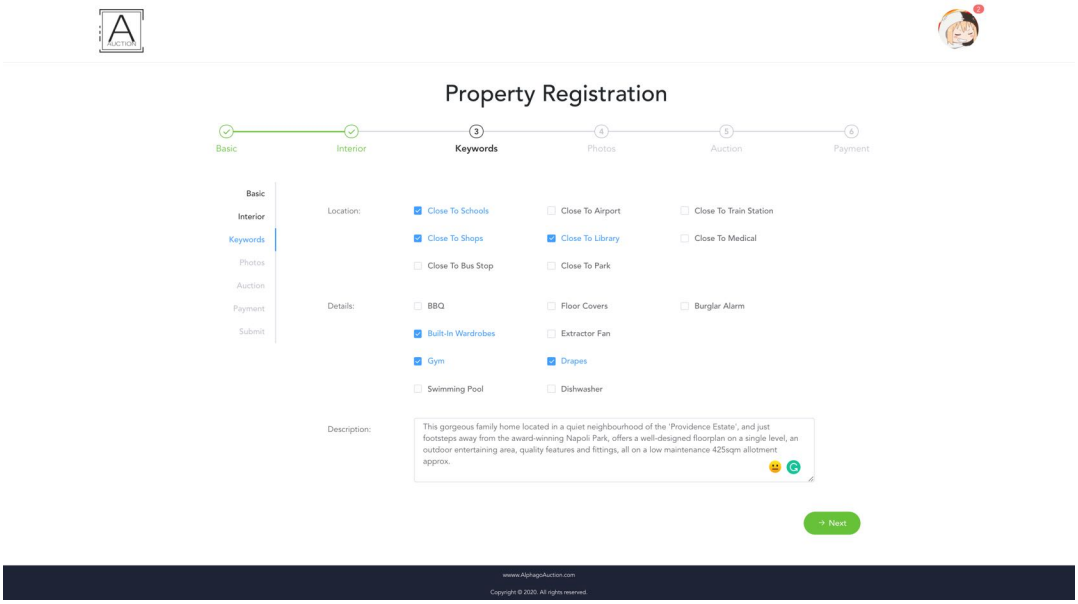


Figure 3.3.2.3.1 Property Registration Section 3

3.3.2.4 Photo Uploading

The fourth part is to upload photos of your property. A limitation of the number of photos uploaded is five. Once a photo is selected, the user is able to delete it by moving the mouse on the photo and click the trash bin button.

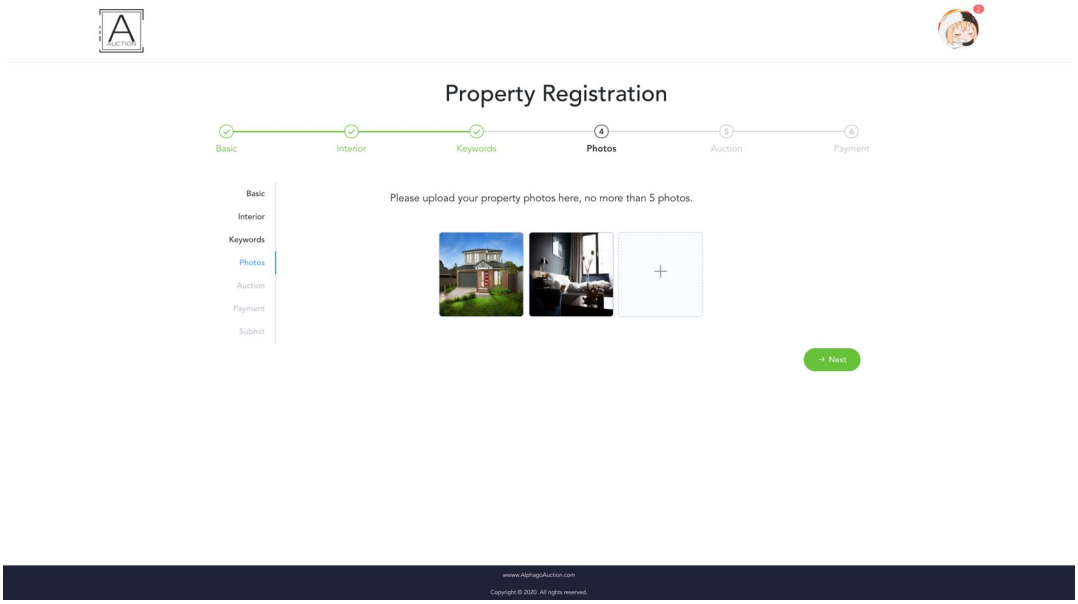


Figure 3.3.2.4.1 Property Registration Section 4

3.3.2.5 Auction Registration

The fifth part is optional. The user can decide after uploading, whether an auction should be registered immediately. If the user switches on the auction button. The auction registration form will show up. Else, the user will be directed to the Finish section.

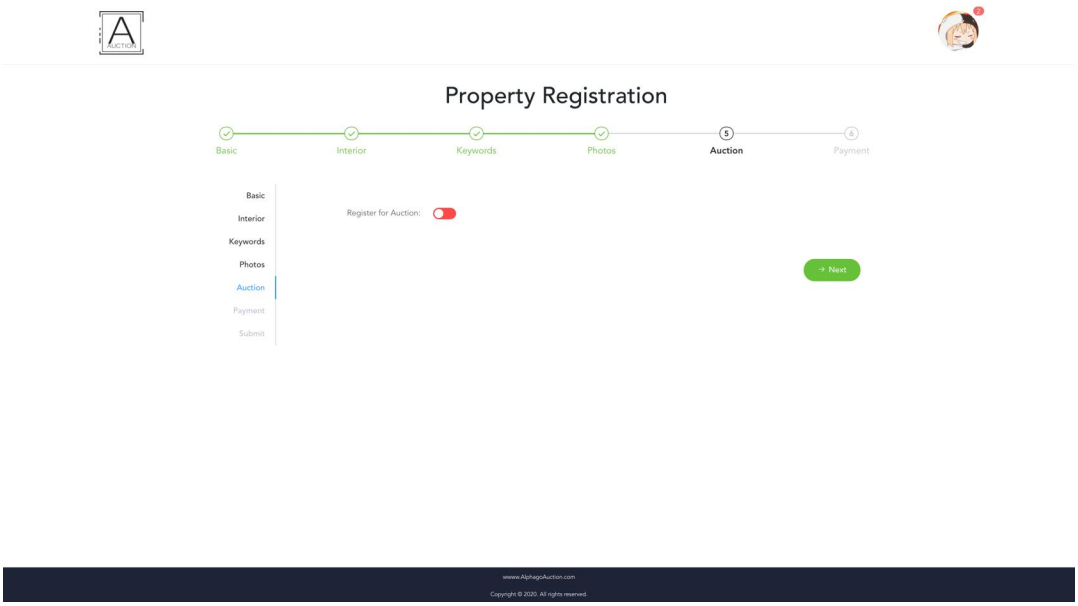


Figure 3.3.2.5.1 Property Registration - Auction Registration Switch

In the auction registration section, auction start time, end time, the reserved price and the starting price are required. Start time should no be earlier than the day that the user registers the property. And the end time should be later than the start time. The starting price should be smaller than the reserved price, and all the price are limited in 11 digits. Additionally, the seller's bank card account is required. The user can either choose an recorded bank card or add a new card for payment method registration.

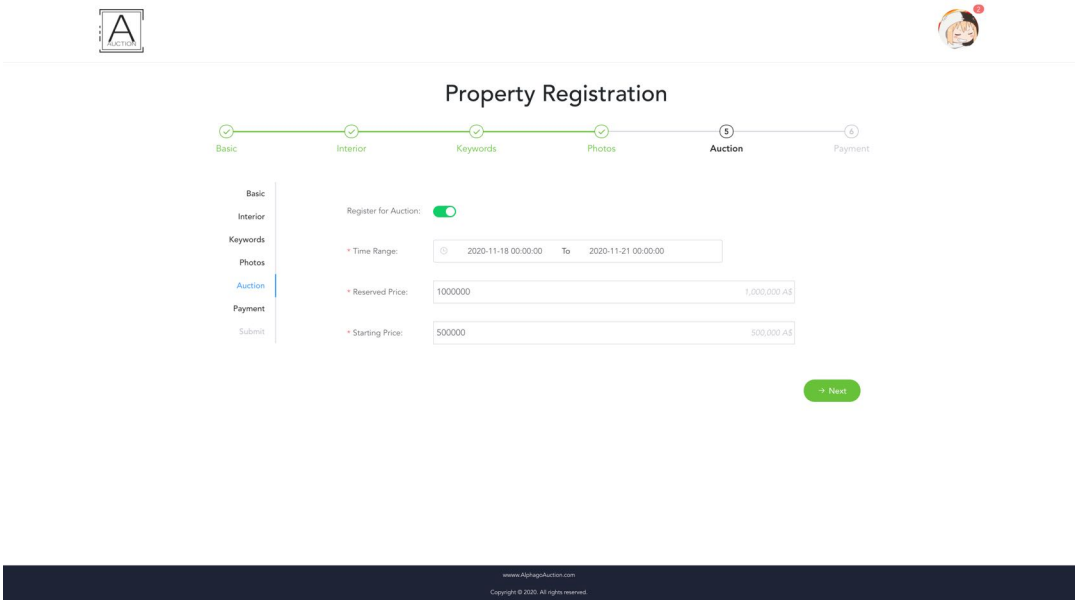


Figure 3.3.2.5.2 Auction Registration in Section 5 - Auction Fundamental Information

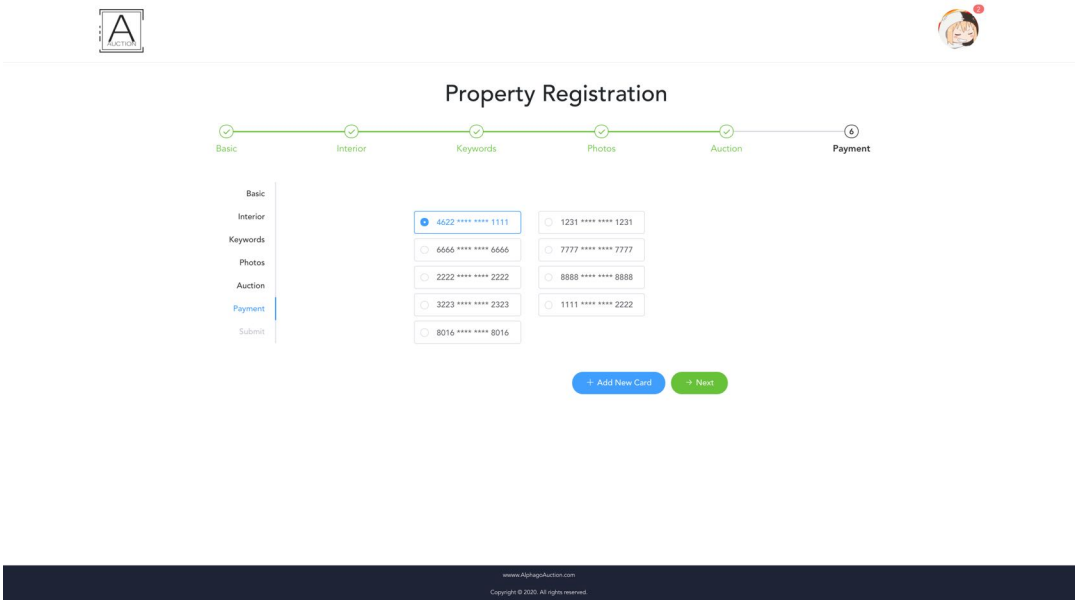


Figure 3.3.2.5.3 Auction Registration in Section 5 - Choose An Existing Bank Card

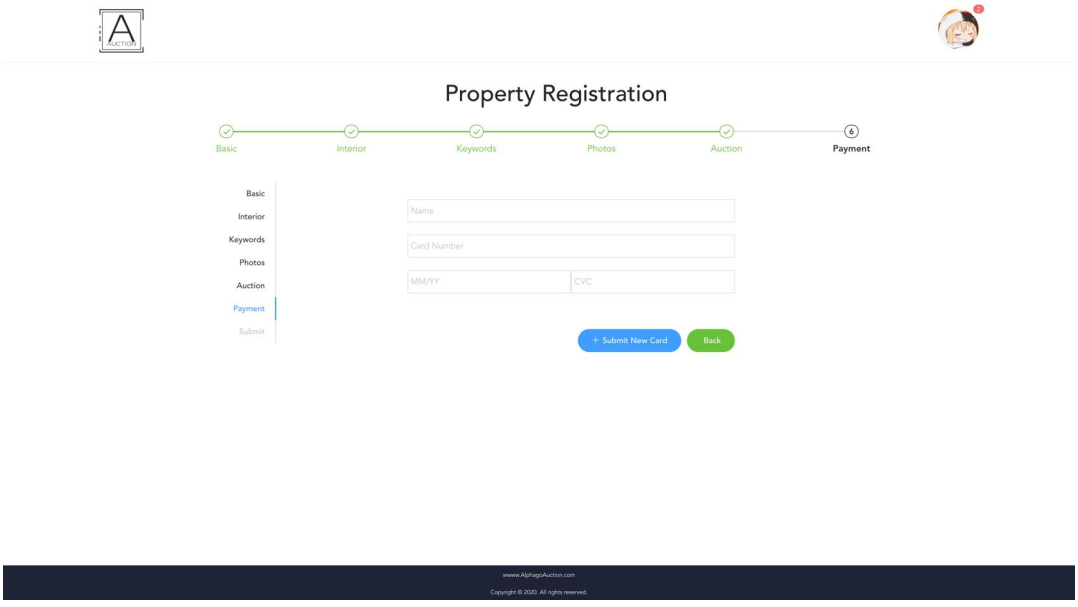


Figure 3.3.2.5.3 Auction Registration in Section 5 - Add A New Bank Card

3.3.2.6 Registration Finish

Once the "Submit" button is clicked, all information will be sent to the back-end and the router will direct the user to the property management page.

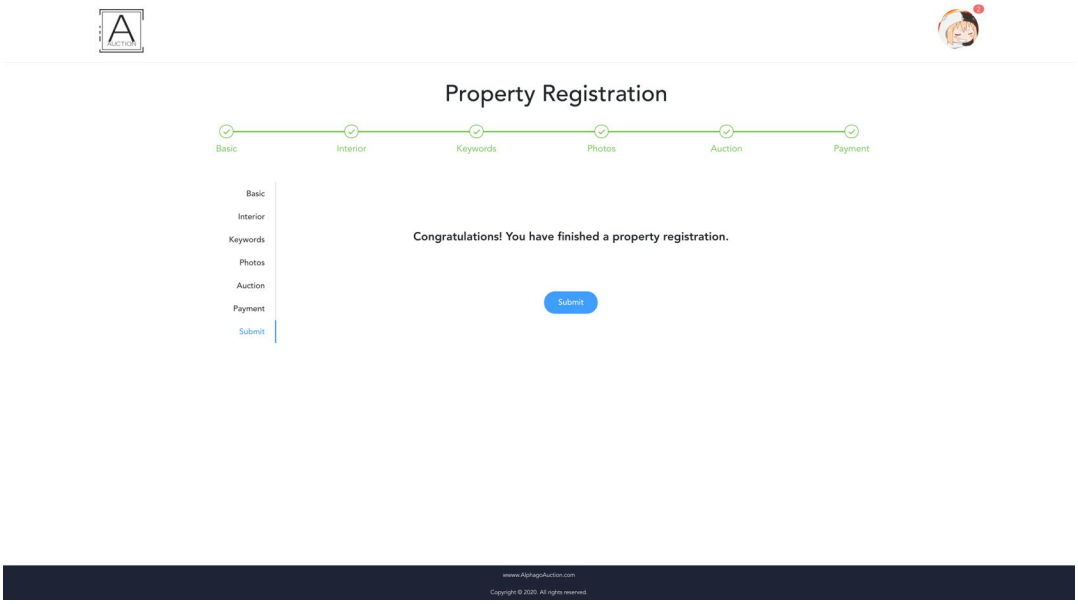


Figure 3.3.2.6.1 Property Registration Done

3.4 Auction

3.4.1 Auction Page

In order to ensure the real-time and uniformity of the auction, on the auction page, we use WebSocket to keep the connection alive between the server and the client while the user is visiting this page. Once a user opens this page, we will establish the WebSocket connection with the server to guarantee the content on this page is dynamically updated in real-time.

If the user is the seller, this page will prompt the user that "You are the seller".

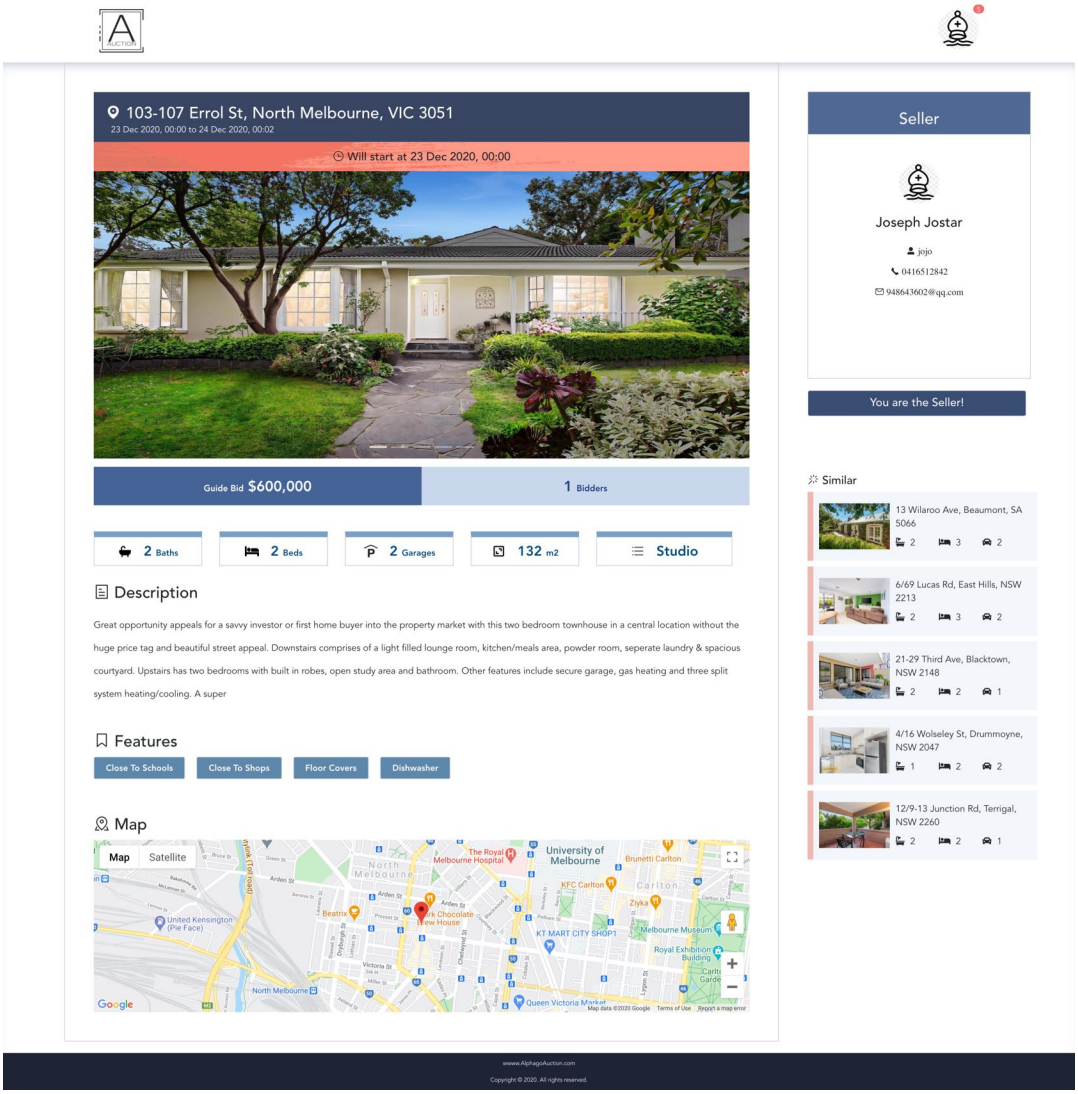


Figure 3.4.1.1 Auction Page

When the auction has not started yet, we use a red banner to show the start time of the auction. At this time, a user can register as a bidder for the first bid. The Auction Page also dynamically updates the number of registered bidders in real-time through WebSocket. When the time reaches the auction start time, we will send a signal through WebSocket to automatically refresh the page.

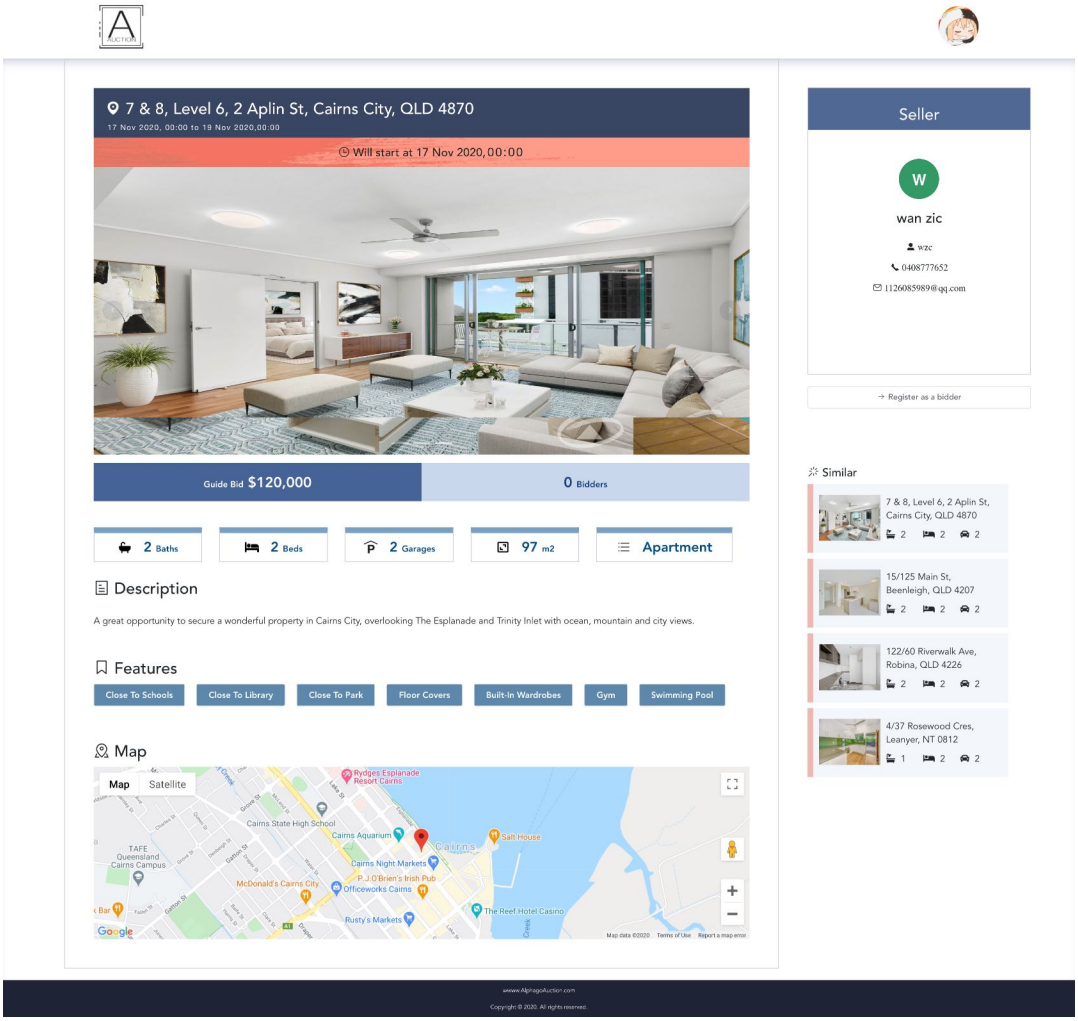


Figure 3.4.1.2 Auction Page - Not Started Yet

If the user has not logged in, the system will first remind the user to log in to the account. At the same time, the router will record the ID of the property that the user browses. After the user logs in successfully, the router will send the user back to the previously viewed auction page.

Once the user has finished the bidder registration process (section 3.4.2), but the auction is still waiting to begin, the user cannot place a new bid until the auction commencement. In this scenario, whenever the user clicks the "Place New Bid Here" button and tries to bid, a message " The auction has not started!" shows up below the initial bid made in the registration process to remind the user.

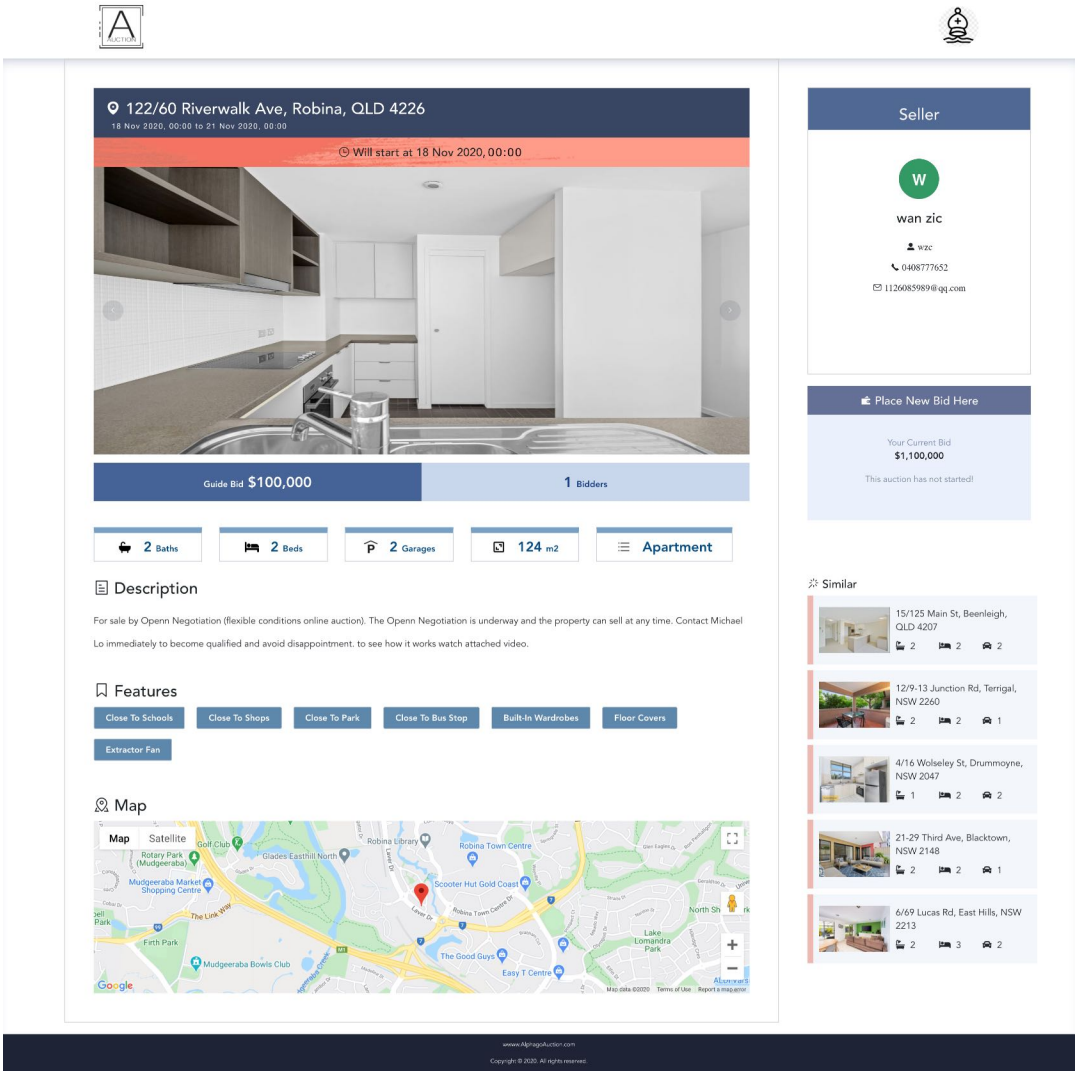


Figure 3.4.1.3 Auction Page - No Bid Access Until Auction Commencement

When the auction starts, there will be a history section on the page. This section is mainly used to record all the bidding history of the property, sorted in a descending order of price, and the record of the temporary winner in the first row will be highlighted.

Whenever someone bids successfully, the system alerts each user who is viewing this page through WebSocket, and the page will have the following changes:

- A pop-up window appears at the upper right corner of the page, which notifies with the latest bidder's username, bidding time, and bidding price. If the user does not close the pop-up window, it will stay for 30 seconds.
- The latest bid price of the property changes dynamically.
- If a new user participates in the auction, the number of bidders is dynamically updated.

- Information about the latest bid is dynamically inserted into the history.
- If a registered bidder makes a bid within five minutes before the end of the auction, the auction end time extends two minutes.

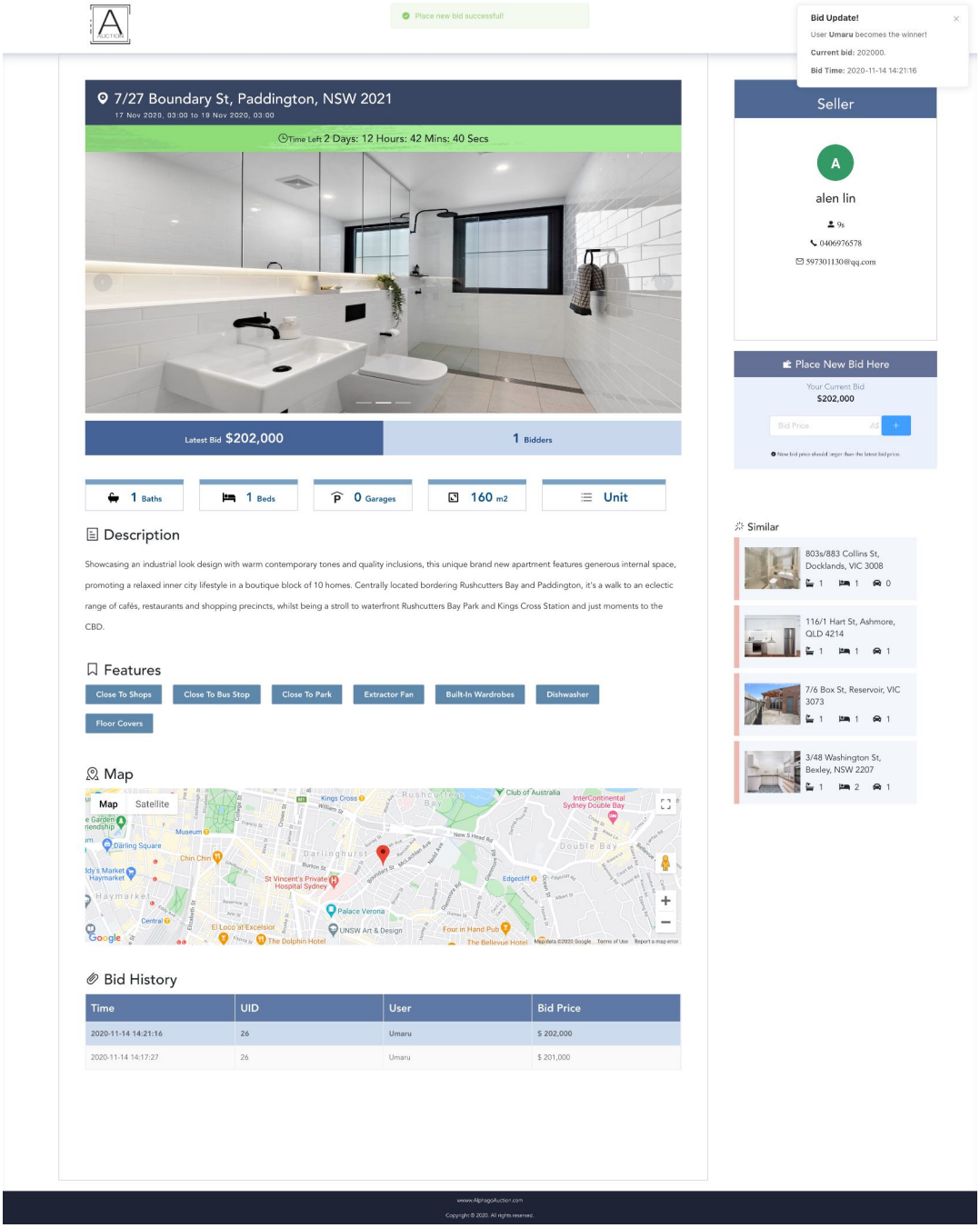


Figure 3.4.1.3 New Bid Created

3.4.2 Bidder Registration

If a user wishes to register as a bidder, the user needs to provide payment information. The user can choose the bank card information that has been submitted before or add a new bank card.

If the user has submitted bank card information before, then all the cards they holds will be listed. Also, the user can click the "Add Card" button to register the payment information, and then click Next to fill in the initial bid, and then submit the registration.

If the user successfully submits the new bank card information, we will redirect the user back to the card selection page, and the user can see and select his new bank card to register.

At the submission stage, the latest guided price shows up, and if the registered user accepts this guided price as his/her initial bid, he/she can click the "submit" button for confirmation. Once a new initial bid is confirmed by the back-end, the guided price will increase by $10, so that there would be no same initial bids.

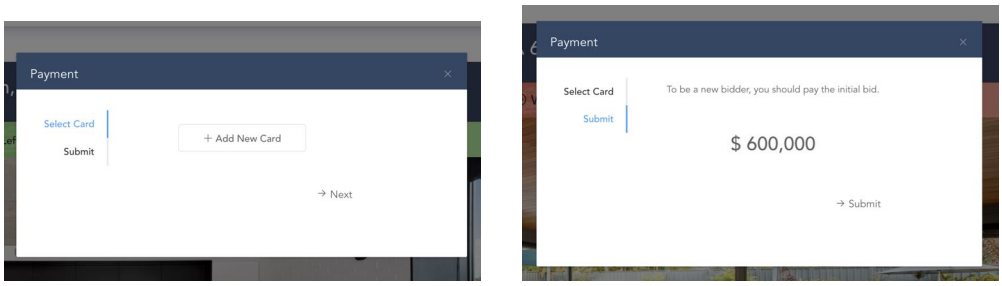


Figure 3.4.2.1 Register as a bidder

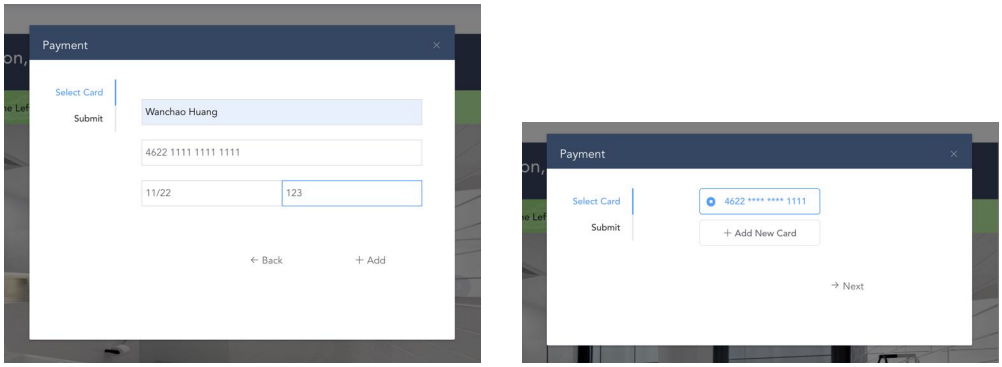


Figure 3.4.2.2 Add A New Card

3.4.3 Scenarios of Unavailability

When an auction ends, the access to this particular auction page should be forbidden, and if a user goes to this page at this moment, he/she will see the page as Fig 3.4.3.1 revealing that the auction has ended already.

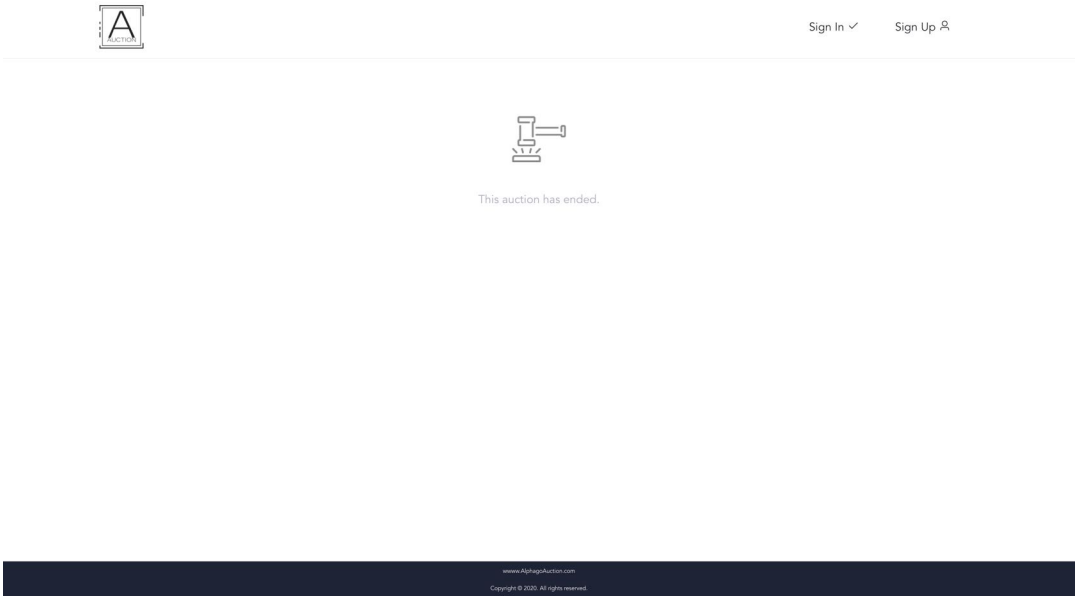


Figure 3.4.3.1 403 Forbidden Page

When it comes to malicious web page visit that a user tries to go to an auction page which does not exist, the user will be redirected to a 404 page given as Fig 3.4.3.2.

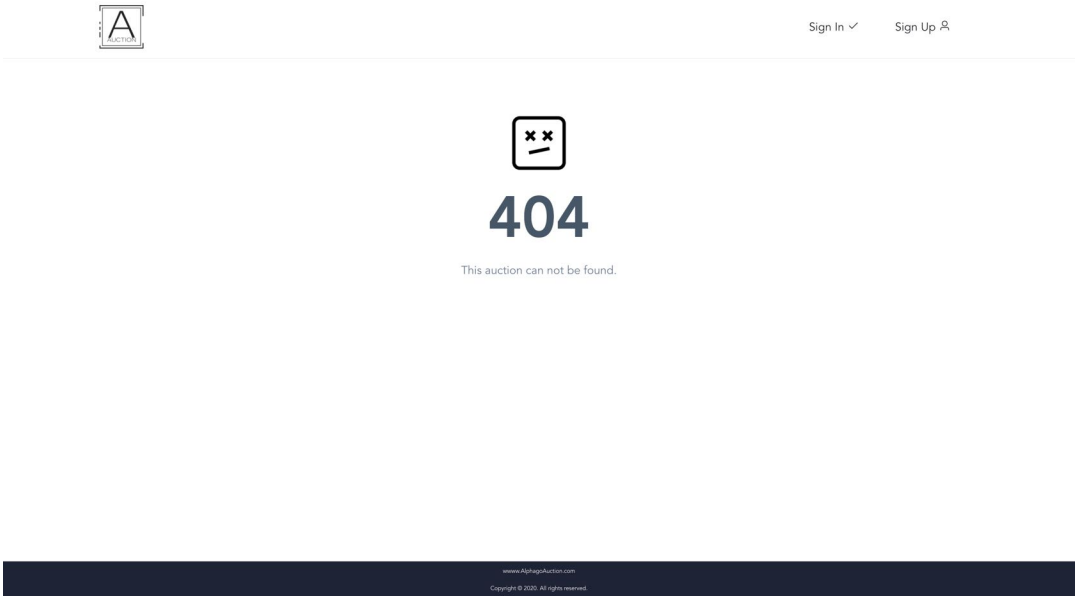


Figure 3.4.3.2 404 Not Fou Page

3.4.4 Auction Management

If a user has no auction records, the system will remind the user on the top.

Figure 3.4.4.1 Auction Management is empty

When you click the "My Auctions" button in the drop-down menu under the avatar, you can jump to the auction management page. This page mainly displays all auctions that the user are participating in.

The bidding here is divided into three different states, namely:

1. Unstarted auctions are marked in red.
2. The bidding has started, but the user's bid is not the highest price, which is marked in green.
3. The bidding has started, and the user is the temporary winner of the current bidding, which is marked in gold.

There is a button to view the history at the upper right corner of the page. After clicking it, user can view the auctions that has participated in and have ended.

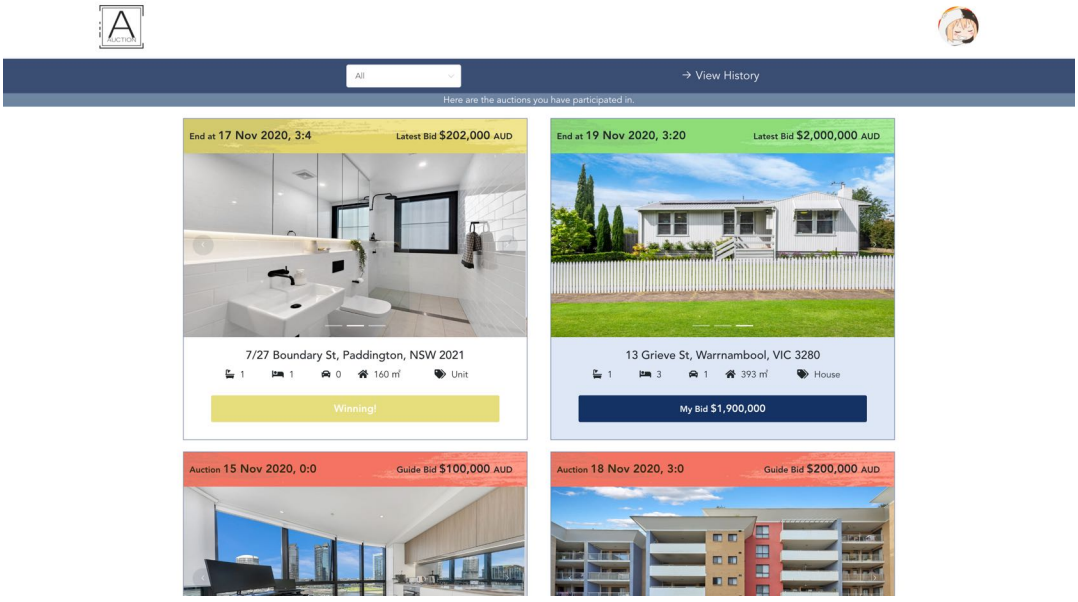


Figure 3.4.3.2 Auction Management

3.4.5 Auction History

If a user has no auction history records, the system will remind the user on the top.

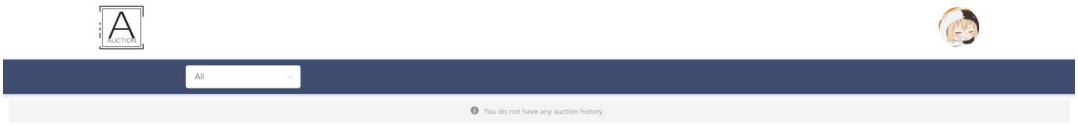


Figure 3.4.5.1 Auction History Is Empty

Otherwise, in the auction history page, there are three states of auction:

1. After the auction is over, the property is successfully sold, but the user is not the winner, marked in green.
2. After the auction is over, the property has passed in, marked in grey.
3. After the auction is over, the property is successfully sold and the user is the winner.

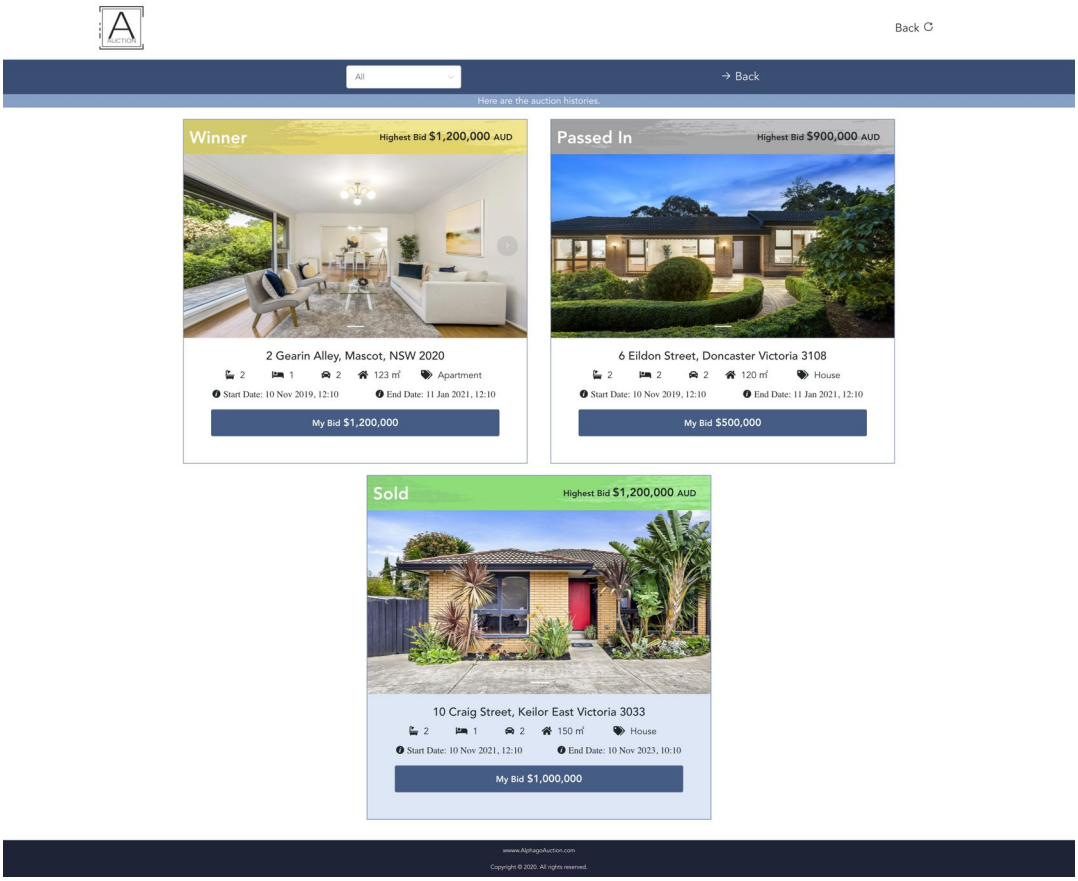


Figure 3.4.4.2 Auction History

3.5 Notification

When the auction is over, we will send notifications to the corresponding users based on the auction information. On all pages where the user's avatar can be seen, there will be a red dot reminder at the upper right corner of the avatar, with a number of unread notifications at its center. If the user reads a unread notification, the number will reduce by one. And the title of the notification will change from bold to light font. If the property is successfully sold, we will send messages to both the seller and the winning bidder. If the property is not successfully sold, we will only send a notification to the seller with the bidding history.

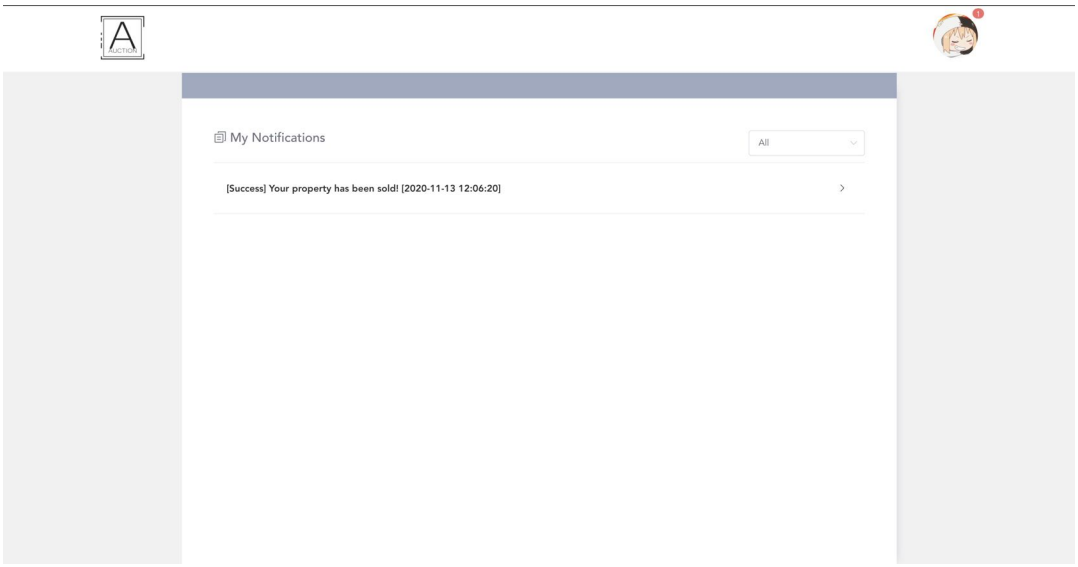


Figure 3.5.1 Unread Notification

The sample message below is send to the seller of the property which has been sold successfully.

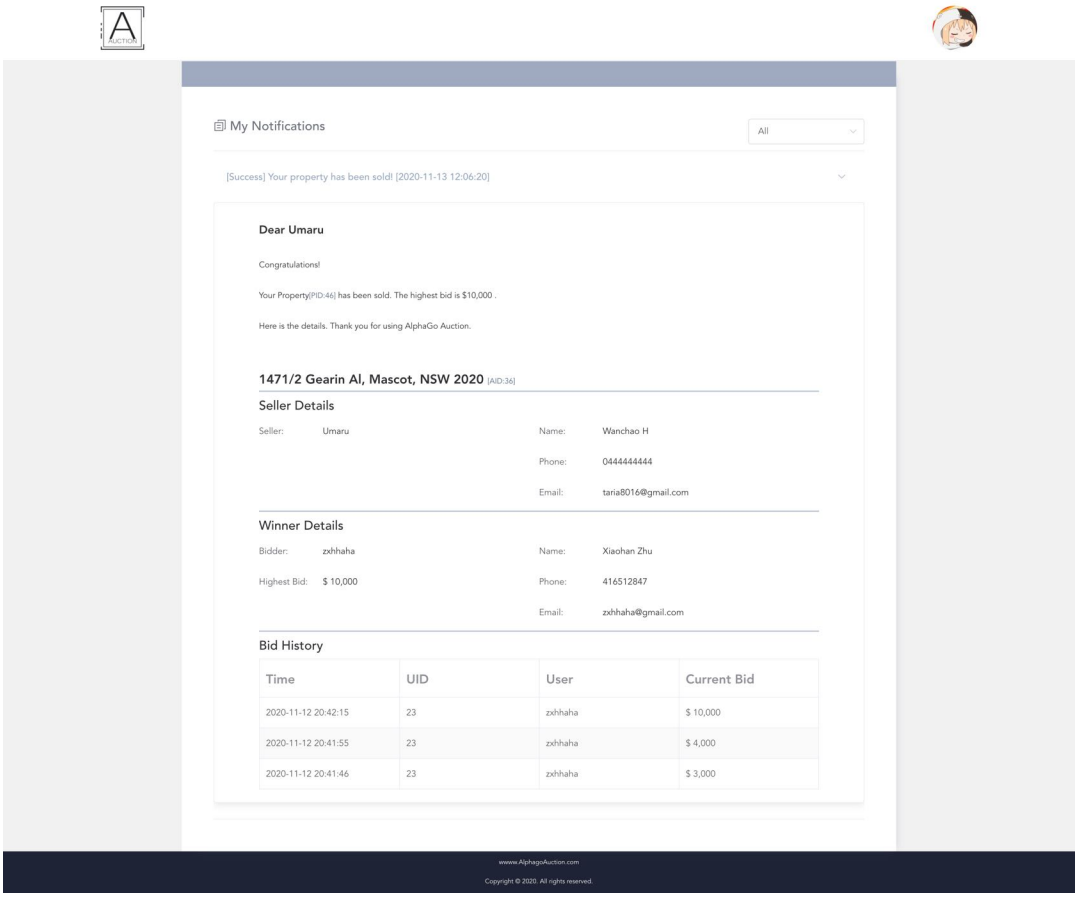


Figure 3.5.2 Seller Notification

The sample message below is send to the winner of the property which has been sold successfully.

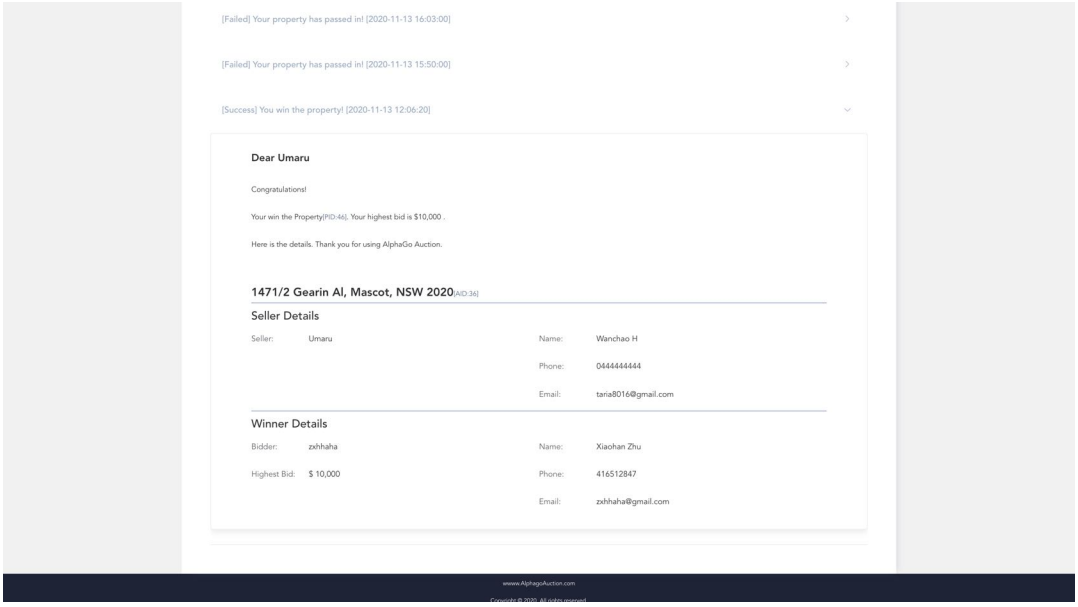


Figure 3.5.3 Winner Notification

The sample message below is send to the seller of the property which has been passed in.

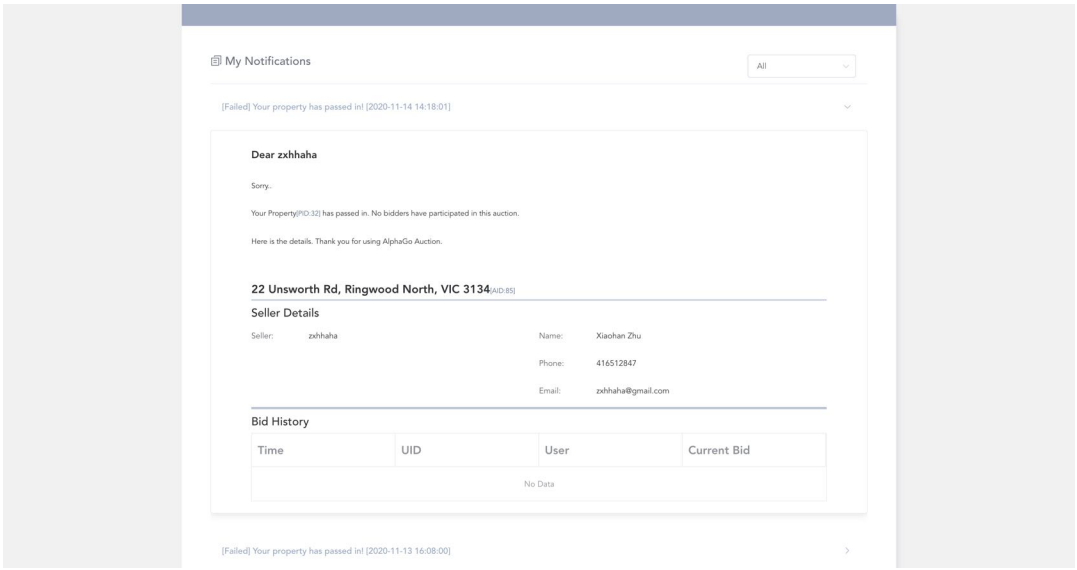


Figure 3.5.4 Failed Notification

4 Third-Party Functionalities

4.1 Vue.js

Vue.js (VueJS, 2020) is a progressive framework that simplifies the construction of user interfaces. The reason why we choose it is that it makes the front-end teamwork easier as it separates a web page application into independent components, and members can develop their corresponding components without heavy reliance on others' code, which has the potential to boost up the development speed. Besides, the syntax of Vue simplifies the management of Document Object Model (DOM) events, from which developers do not need to write complicated JavaScript functions and there are less risks of JavaScript function collision. Moreover, class and style binding, list rendering, and event handling using *v-on* from Vue make the front-end well structured and easy reading.

Vue.js is licensed under the MIT license, and is granted, free of charge, to any person obtaining a copy of this software and associated documentation files by the author Evan You. Its copyright is from 2013 - present.

4.2 Elements and Components for Front-End

In industry web developments, encapsulated components are widely applied for saving time and better appearance.

4.2.1 Element UI

Element UI is a Vue 2.0 based component library (ELEMENT-UI, 2020), from which this project utilizes components like calendar, pagination. We choose Element UI because it provides various universal components that can save our time from creating code snippets repeatedly. Also, the components are stable so we do not need to worry about bugs in their functionalities.

Element UI is licensed under the MIT license, and is granted, free of charge, to any person obtaining a copy of this software and associated documentation files by ElemeFE. Its copyright is from 2016 - present.

4.2.2 Vuetify

Vuetify is another good library of Vue components which aims to provide developers with plenty of choices of components, and users with great interactive experience (Vuetify, 2020). We choose the card component from it to display the search results as we assume the card is of aesthetics.

Vuetify is licensed under the MIT license, and is granted, free of charge, to any person obtaining a copy of this software and associated documentation files by the author John Jeremy Leider. Its copyright is from 2016 - 2020.

4.2.3 FontAwesome

Font Awesome (FontAwesome, 2020) is a website which provides free icons. To use icons to beautify our pages, we copy the link to its CDN at the *index.html*.

FontAwesome Free is licensed under the MIT license, and is granted, free of charge, to any person obtaining a copy of this software and associated documentation files.

4.3 Spring Boot

Spring Boot (Spring, 2020) is a simplified Spring framework, which allows users to construct applications in a fast way. As this project is comparatively simple, and back-end members are familiar with Java, this framework is chosen. Moreover, a light framework as the Spring Boot is, the advantage of fewer configuration and small size can enable the project to fit the VLab platform better.

Spring Boot is licensed under the Apache License 2.0, and is granted with a perpetual, worldwide, non-exclusive, royalty-free, copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, sublicense and distribute its Work and any resulting derivative works in any form.

4.4 Google Maps Platform

Google Maps Platform (Google, 2020) provides various tools that can realize complicated features in web applications. In this project, Google Maps JavaScript API, Places API and Geocoding API are utilized. On home page and search page, the searched suburb names will be validated by Google Maps JavaScript API before passing the value to the back-end. On the page of property registration, all three APIS are applied to the address-auto-complete functionality for finding existing addresses and converting addresses into geographic coordinates. With geographic coordinates and supports from the Google Maps JavaScript API, on the property auction page, the *vue2-google-maps* package can display the neighbourhood of the on-sale property.

Google Maps Platform is licensed under the Google Maps Platform Terms of Service, and is granted to customer a non-exclusive, non-transferable, non-sublicensable license. AlphaGo team pays for the Google Customer Services.

4.5 Amazon S3

Amazon Simple Storage Server, namely Amazon S3 (Amazon, 2020) is an ideal platform for static data storage with the merits of reducing reading and writing to back-end databases. Also, trustworthy as Amazon S3 is, there are less worries for image loading failure due to S3's 99.99% data durability (Amazon, 2020). And S3's safety against the high-frequency ransomware attack is better than traditional databases, with a traditional database has 16.2 days of downtime per year on average (Gill, 2020). Furthermore, the front-end can load images faster from Amazon S3 than MySQL.

Amazon S3 is licensed under the Amazon Software License, and is granted to customer a a perpetual, worldwide, non-exclusive, royalty-free, copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, sublicense and distribute its Work and any resulting derivative works in any form.

5 Implementation Challenges

5.1 Front-End Part

5.1.1 User Status in Front-End

5.1.1.1 Problem Statement

In modern web applications, users do not need to input their credentials multiple times for login. This practice gives users better interaction experience and saves users' time. Moreover, in this project's scenario, if there are a considerable number of users login within a certain time period, the workload for database will be tremendous. As designers, we should endeavour to minimize the burden on database for good hardware maintenance.

5.1.1.2 Solution

The solution we apply is using *Vuex* and *localStorage*. *Vuex* is a state management pattern plus library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion. Since Vue.js is a single page application tool, it cannot have a centralized actions or mutations for all of its pages. *Vuex* solves this problem by mapping, and with it, state information for the whole web application could be stored and obtained easily. On individual Vue pages, we use *mapActions()* functions to direct the functionalities to the *index.js* file under the *store* directory which is specifically for *Vuex*. By doing this, the state information could be unified.

As for the *localStorage*, it is a read-only property which allows you to access a Storage object for the Document's origin. In contrast to other storage properties such as *Cookies* and *sessionStorage*, *localStorage* has the advantage of long-term memory meeting the requirement that users do not want to repeatedly login over a period of time, while, *Cookies* has the risk of data loss once the browser is closed. After the first time login, the system saves the JWT token, user name, user's first name and user's avatar into the *localStorage*. Since that, every time the user is linked to AlphaGo Auction, the user information will be loaded from the *localStorage* before the web page is completely rendered.

5.1.2 Style Bugs in NPM Package

5.1.2.1 Problem Statement

When we applied the *vue-better-google-places-autocomplete* package, there was a bug in its CSS style that its inherent JavaScript overwrites the pre-defined CSS style in the Vue files when pages were being rendered, and there were dots before the tags. Attempts on using JavaSript or JQuery both failed to change the inherent CSS styles in this package.

5.1.2.2 Solution

Eventually, we found that using " !important " keyword can provide more weights to the chosen CSS style, and no other normal CSS style can overwrite it. In later practice, when it came to the collisions between CSS styles between different pages, " !important " was a helpful tool for style management.

5.1.3 Address Validation

5.1.3.1 Problem Statement

In industrial property auction applications, a property's address needs to be validated before registration in case of malicious scams. We wanted to enable our web application to have this capability of validation. However, only given 9 weeks for development, it is unlikely for the team to make up a geo-location validation functionality.

5.1.3.2 Solution

Google Map is a well-known map platform for people to search addresses. By utilizing Google Maps Platform API, web applications have access to the Google Map services. Google Maps JavaScript API only provides sample codes run in pure JavaScript environment, which does not fit in Vue.js framework. Three packages named *vue-better-google-places-autocomplete*, *vue-google-autocomplete*, and *vue2-google-maps* are used to integrate the Google Map services into this project, with the first one, whose CSS styles could be easily changed, is used for address validation in the property registration; the second one is applied to display property neighbourhood on the auction page; and the last one, which has an option to restrict its search scope, is utilized to validate suburbs in search modules.

5.2 Back-End Part

5.2.1 Ransomware Attacks

5.2.1.1 Problem Statement

During our development, we had gone through ransomware attacks to our MySQL database for twice. The ransomware removed all of our data and tables, and asked for Bitcoins to blackmail.

5.2.1.2 Solution

Since we have no accesses to improve the security of MySQL, we could only backup our data and schema, though it took us a whole night to recover. After making backups, we reset a more complicated password to reduce the risk of getting attacked and reported this situation to tutor.

5.2.2 Design of Recommendation System

5.2.2.1 Problem Statement

The design of recommendation system is the novelty in this web application and this system is designed upon the search history and bidding behaviours of a user. In the process of designing the recommendation system, we need to quantify the attributes of a property, so that we can have ways to find similar properties later. The quantification of the numbers of bathroom, bedrooms, and garages is comparatively easy as they are all numerical. Nevertheless, how to find properties that are closed to historical searched real estate, is associated with the distance calculation, and Google Maps JavaScript API could not help in this scenario since the back-end is using Java programming language.

5.2.2.2 Solution

After deliberation, we decided the main idea of our recommendation system is "calculating the average data for each user's behaviour. For detailed description of user historical behaviour, please refer to section 2.2.2. When the system tries to recommend a user with some properties, it looks into the **HISTORY** table to extract the user's behaviour data, and creates a vector describing the user's history as followed: $<$ $bedroom\_num/bedroom\_cnt, bathroom\_num/bathroom\_cnt, garage\_num/garage\_cnt,$ $lat/lat\_cnt, lng/lng\_cnt >$. Then, the system traverses the **PROPERTY** table and summarises the features of each property as the vector $< bedroom\_num, bathroom\_num,$ $garage\_num, lat, lng >$. After that, the Euclidean distances between the vector from **HISTORY** table and each property's vector are computed, and the overall computation results are sorted in a ascending order. It is known that the shorter Euclidean distance indicates higher similarity between two vectors. In this case, a property with a smaller Euclidean distance from the query vector from the **HISTORY** table means it is closer to the user's favor. Eventually, the first three properties with the lowest Euclidean distance are recommended on the auction and search pages.

5.2.3 Novelty in Synchronization

5.2.3.1 Problem Statement

In real-time auctions, we hope users could be notified with the latest bid price and the current highest bid price on the auction page could be synchronized with the latest one. This could be represented in the scenario that when multiple users are browsing the same property auction page, whenever one of them makes a bid, the other users could be notified with the latest highest price synchronously. However, contemporary property auction websites do not have this feature.

5.2.3.2 Solution

In order to realize the synchronization, we need a communication protocol between the front-end and back-end that can guarantee the back-end can keep sending the latest

information to the front-end without front-end page refreshing. Under this scenario, the WebSocket protocol is introduced. Every time a certain property's auction page is rendered, a WebSocket connection to the back-end server is established, and this connection will not be destroyed until the user closes the current auction window. Even if an issue happens in the connection, the front-end page will initialize another WebSocket connection to the back-end. Every time a registered bidder makes a bid on a property auction page, the latest bidding will be passed to the back-end server, and then be propagated to all front-end pages by the server. The new bidding will be displayed in the format of both a floating notification window and the updated current highest bid on the auction page. By this mean, there will be less probability of collisions between bids such as a newly created bid is smaller than a higher one which was made few seconds ago.

6 User Manual

For functionalities of this web application, they are well demonstrated in the section 3, and we do not repeat the similar contents again in this section.

NOTE: This web application is only working on 16 inch screen monitor. Smaller size monitor is incompatible now.

6.1 Front-end Setup

Steps to set up the front-end:

1. Change directory to “front-end”.
2. Set up the npm environment. There is a txt file in our frontend folder named ”node_modules_link”. If the user’s npm version is too old, they can download the node modules via the link in this file.

   Once you download the node modules zip file to the frontend folder. Double click the zip file and extract the “node_modules” folder to the current path. And after the extraction, the folder content is as shown in Figure 6.1.2.

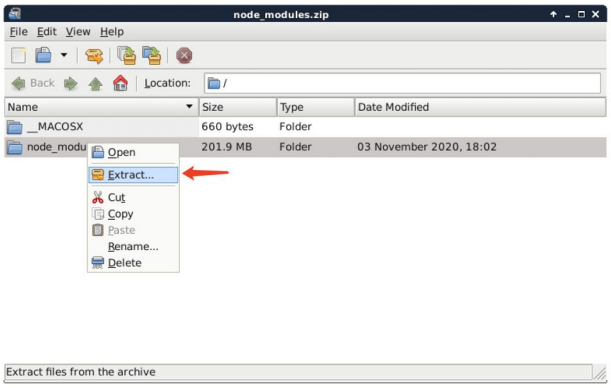


Figure 6.1.1 Vlab

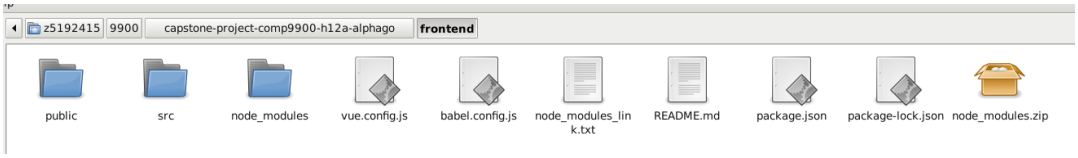


Figure 6.1.2 Vlab Frontend folder

3. Run the command in the console in this folder:

   `npm run serve`

4. Then the terminal is supposed to show the content below, and now you can visit our website on

   "http://localhost:8080/"

Figure 6.1.3 Front-end

6.2 Back-end Setup

- The system must be run under JRE 8+.
- The system must has Maven.

  Both things are provided in VLAB environment.

6.2.1 Building System

We have already provided the packed server: **propertysale-0.0.1-SNAPSHOT.jar** in the root directory. Alternatively, you can build the server by yourself following steps below:

- Click to enter the *propertysale* folder where is source file of the system, where you can find **pom.xml**.
- Run the Maven command to pack the server in the console:

  `mvn install -f pom.xml`

- Now, you can find the packer jar file **propertysale-0.0.1-SNAPSHOT.jar** in the **target** directory.

6.2.2 Running System

- Change directory to the **target** directory.
- Run *Java* command under the directory which has **propertysale-0.0.1-SNAPSHOT.jar**:

  ```
  java -jar propertysale-0.0.1-SNAPSHOT.jar
  ```

  Now, you can find the system is running on Port 8060.

6.3 Amazon Web Server Recovery

We have already started a cloud service on Amazon Web Services where we are running database, Redis and other important components in our system. However, the AWS is very unstable that it may crash. If the server cannot connect Redis or RabbitMQ, which means it need to be restarted. So that, you should follow the steps below to recovery the cloud server:

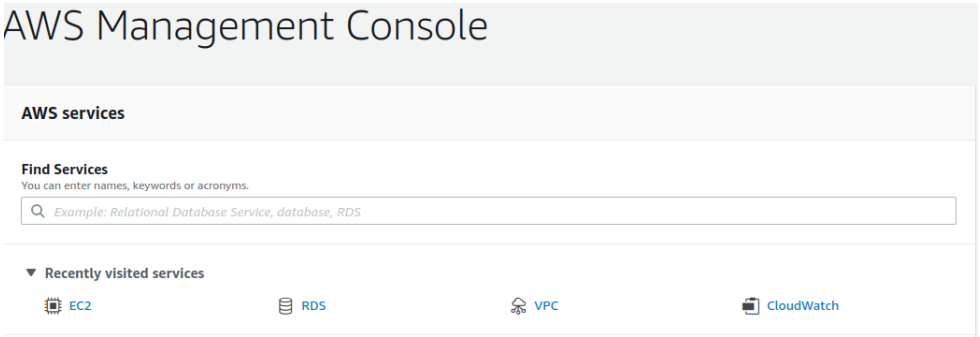
6.3.1 Login

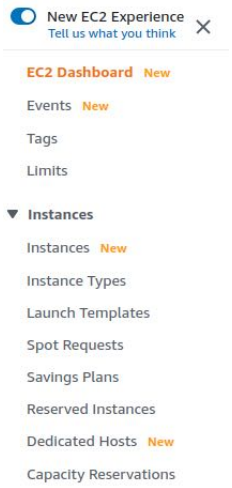
Go to the home page `https://aws.amazon.com/` and sign in with:

Username: alphago9900@gmail.com
Password: Alphago9900,.

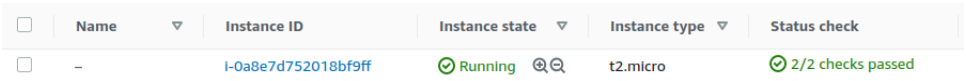
6.3.2 Connecting to Service Instance

- Click **EC2** then select **Instance** on the side bar:

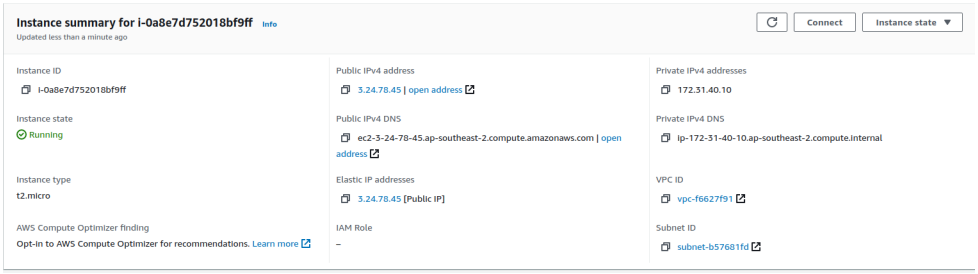




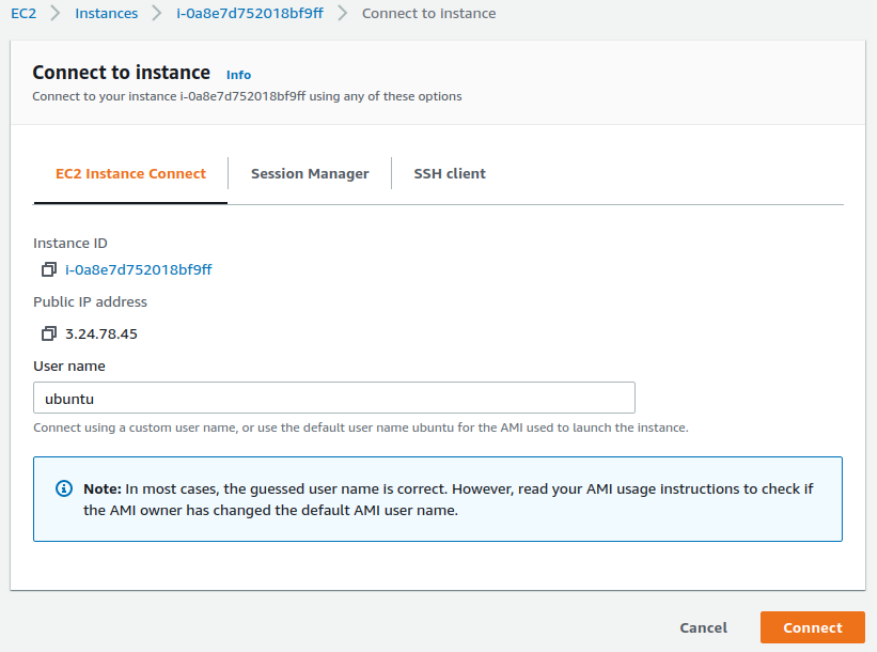
- Click the instance with ID i-0a8e7d752018bf9ff.



- Click the button in the upper right corner.



- Click the orange button connect to connect the cloud server.



Now, we can use console to restart all services:

6.3.3 Restarting all services

Attention!! The following steps must under the ~ or /home/ubuntu. **Do not move to other paths!**

- Restart Rabbit MQ:

  `sudo docker restart rabbit`

- Restart Redis:

  `sudo /usr/redis/bin/redis-server /usr/redis/bin/6379.conf`

- Restart Image Server:

  `nohup java -jar /home/ubuntu/ImageServer/security.jar > log.file 2>&1 &`

If the restart is not successful or there are other problems, please contact to Xiaohan Zhu via **z5187021@ad.unsw.edu.au**.

References

Amazon. (2020). *Amazon s3 - object storage built to store and retrieve any amount of data from anywhere.* Retrieved from https://aws.amazon.com/s3/

ELEMENT-UI. (2020). *Element ui design disciplines.* Retrieved from https://element.eleme.io/ #/en-US/ guide/design

FontAwesome. (2020). *Fontawesome basic use.* Retrieved from https:// fontawesome.com/ how- to-use/ on- the-web/ referencing -icons/basic-use

Gill, M. (2020). *10 shocking data loss and disaster recovery statistics.* Retrieved from https://www.comparitech.com/ data-recovery-software/ disaster-recovery-data-loss-statistics/

Google. (2020). *Google maps javascript api v3 reference.* Retrieved from https://developers.google.com /maps/documentation/ javascript/ reference

Kotha, S. (1995). Mass customization: Implementing the emerging paradigm for competitive advantage. *Strategic Management Journal 16 (S1):21–42*. doi: http://dx.doi.org/10.1002/smj.4250160916

Lee, L., & Adrian, S. (2017). *SELING REAL PROPERTY: Benefits of online auction platforms.* Retrieved from http://search.ebscohost.com/login.aspx?direct=truedb=buhAN= 123350652site=ehost-livescope=site

Spring. (2020). *Spring boot overview.* Retrieved from https://spring.io/projects/spring-bootoverview

Tan, S.-L. (2017, May). *$20 billion in "proptech" globally by 2020: Taronga and KPMG.* Retrieved from https://www.afr.com/property/20 -billion-in-proptech-globally-by- 2020-taronga-and-kpmg-20170523-gwb6h6

VueJS. (2020). *Vue.js introduction.* Retrieved from https://vuejs.org/v2/guide/

Vuetify. (2020). *Vuetify introduction.* Retrieved from https://vuetifyjs.com/en/introduction/why-vuetify/