CO600 Project Report

Property Manager - Android application for Estate Agents

EW315@kent.ac.uk
AP712@kent.ac.uk
Ol61@kent.ac.uk
Ohaa2@kent.ac.uk

School of Computing
University of Kent
United Kingdom

Aaron Pedwell: 2610 Words
Elly Winder: 2661 Words
Obaid AlMansoori: 252 Words
Othman Lijji: 346 Words
Final Word Count: 5869

Date: 29th March 2019

1. Abstract

We will be creating a mobile application that aims to speed up the time taken by an estate agent to prepare property details. The current system used by estate agents is mainly paper based with the agent attending a property to make written notes, taking photos and measuring the room sizes. The aim of our application is to provide these functions within a simple mobile application. This would allow users to record information regarding properties with the aid of dropdown menus to minimize the time spent at the property. The application will provide the ability to take photos, which will then later be used to create a listing based on a template via a website.

2. Introduction

Online competitors are chipping away at high street agents profits by providing a service that is easier to use, more efficient and are easily accessed. By having someone in our group who works in a local estate agent we asked her and her colleagues what difficulties they find in their job. One of the key issues was the time and effort it takes to add new properties onto their advertising sites. Most agents currently use a paper system, with separate databases. When they need to add a new property to their database, they must manually input the information in the office, following a visit to the property to gather the details and photograph. Property Manager aims to simplify the process from start to finish by allowing the user to input all this information into an application whilst at the property and taking the photographs in one, easy task. This data is saved to a database for editing/viewing later via the website. The website also automatically generates an advert for the user to aid in advertising the property on all platforms.

3. Background

3.1 Market Research

The UK property industry has an estimated revenue of £11.1bn a year (Ibisworld.co.uk, 2018) with an annual growth of around 1%. It employs an estimated 158,619 members of staff through 20,823 businesses (Ibisworld.co.uk, 2018). Statistics show that there were approximately 51,000 estate agents and auctioneers in the UK in 2018 (Statista, 2018). The growth rate of the market may not be very attractive at only 1% but with an estimated revenue of £11.1bn 1% equates to a steady increase of £111m annually which is very attractive.

3.1.1 Thesaurus

This is a desktop application currently used by many estate/letting agents. This holds most of the features we are looking to create. At the time of starting our project this was limited to a desktop application, however they have now introduced a mobile application in 2019. This allows the user to add properties and customers to the database, upload photos and the final advert is sent directly to the company's website. This programs appearance is very outdated and old. (Thesaurus.org.uk, 2019)

3.1.2 InventoryBase

This is an inventory application. Although this application doesn't have the same purpose as what we are going to be doing, it does have some similar features. This allows you to add a property to a database, logging specific details such as the address, how many bedrooms etc it has. This application then allows the user to create rooms and take photographs on the devices camera and finally an inventory is generated. This application is used by letting agents to generate inventories of the properties, rather than a final advert but this is similar to our goal. This application looks very modern and is a similar style to how our project will be. (InventoryBase, 2019)

3.1.3 Handwritten Notes

Currently a lot of estate/letting agents have to visit properties and make handwritten notes and take photographs. They will then go back to the office and correlate this together on a desktop system. This will take a lot more time than being able to do it all whilst at the property on a mobile application.

3.2 Technology

When we initially decided to implement the idea for the Property Manager application, we were planning to use the Mendix platform (Mendix, 2019). However, after initial testing it became clear that although Mendix made it very simple to create user interfaces with simple drag and drop functionality. It was extremely difficult to implement code under the hood. We struggled to find correct documentation to assist us as a new version of their software had recently been released and a lot of the tutorials provided linked back to the old documentation. In the end I researched other options and after creating some basic test apps from tutorials it was agreed that we would move our development to Android studio. This did however limit us to creating the app solely for the Android market base, but we felt this was the correct choice as it uses the Java language that we should all be familiar with. There was the option to use Flutter a new language developed by Google that allows you to create an app using the Dart coding language that will run on both Android and IOS, but this would have required the team to learn a new language and it was felt this may be a serious risk to the project. Aaron then found details of Google Firebase (Firebase, 2019) which provides a full api to handle user registration and authentication for web and mobile applications and access to Cloud Firestore (Firebase, 2019) a flexible database created for mobile and web applications. It provides NoSQL hierarchical storage consisting of collections that contain documents and can also contain nested documents. This was perfect for the needs of Property Manager as we needed to be able to store the customer and property details.

4. Requirements

4.1 User stories

We designed user stories during the early stages of the project. The stories were used to help us picture the end goal for the application and allow us to build the website and application with this in mind. The user stories include:

- An estate agent wants to add a new customer
    - *As an estate agent, I want to add new customers to the database, so that we can store their details and properties.*

- An estate agent wants to add a new property
    - *As an estate agent, I want to add a new property to the database so that it can be viewed on the app and website.*

- An estate agent wants to view a property
    - *As an estate agent, I want to view a property so I can easily see details and who it belongs to.*

- An estate agent wants to view a list of properties
    - *As an estate agent, I want to view a list of multiple properties, so that ...*

We found that user stories were useful during development. At each meeting we could regroup and discuss and analyse the progress for each story. These can all be viewed in the corpus.

4.2 Use Cases

The use case for our application illustrates the functionality required for the system. Since the overall aim of the application is to simplify a process it was important to us that we planned to keep the amount of internal and external influences to only the minimum necessary. This helped us gather the requirements of the system and to show the interactions between the cases and actors. These can be viewed in the corpus.

4.3 Functional and Non-Functional Requirements

4.3.1 Functional Requirements

Android Application

- Add customers & properties – The user should easily be able to locate the add customers or properties button and add a customer. Once created they should see it on the list quickly.
- Edit customer & property details – The user should easily be able to locate the edit button and edit any details required.
- Delete customers & properties – The user should be able to locate the delete button easily and delete a customer easily.

- Link property to customer – The user should be able to link a relevant customer and property together.
- Take photos – The user should be able to use the devices camera to take photographs and save them to the database to be used in the final advert.
- The user should be prompted to add in any missed data to ensure all is correctly input.

Website

- View and add customers and properties – the user should be able to view and add properties or customers easily.
- View advert of property based on data in the database – The user should be able to easily find the 'View Advert' button and generate an advert for that specific property.

Appearance – Applies to both the Website and Mobile Application

- All pages on app & website should use the same shade of blue.
- All buttons should be the same.
- All fonts the same as the logo.
- The Logo should be on all pages of the website and the log in screen of the app.
- Any navigation bars should be consistent.

4.3.2 Non-Functional Requirements

The user should be able to use the application and website without any prior training. They should be able to create a property using the step by step guide easily to ensure no details are missed.

- All data stored should be retrievable in an acceptable time and all pages should load in an acceptable time frame.
- All user data should be kept secure.
- The app needs to allow for high usage during weekdays due to the nature of Estate Agents work, therefore any downtime for maintenance should be done out of office hours or at weekends.
- Users need to be forewarned of any downtime with acceptable notice to ensure they are not adversely affected.
- The mobile Application & Website should pass any testing needed to be made public and distributed.

5. Designs

5.1 Design - Android Application

During our early meetings we discussed the style of the application and it was decided that we wanted to keep it simple, and straightforward. our main priority was ensuring the application flowed in a simple step by step process, ensuring all data is input into the database and nothing has been forgotten. We completed a few Lo-fi designs, which were mainly for the purpose of getting the flow of the application correct, rather than specific design. We did keep to this design for the majority of the application, which showed large, rounded buttons with a plain background, however during production we did need to change some areas as we found that there was too much on the screen and it

was overcrowded, and we felt it would be confusing for the user. Our lo-fi designs can be seen in the corpus.

We then discussed the colour scheme for the application, and upon looking at other, similar types of application we decided to use Blue, as this was commonly used. we felt this looked more professional, than colours such as pink or orange. We also discussed using red, however we thought this may look too bright, and wanted to keep the style more subtle.

We then brainstormed several names for the application, with our initial favourite being "Lister". Which, we felt represented our applications main feature of property advert listing. We then conducted some research into the name, by asking several people what they thought the application would be for when hearing the name for the first time. The main feedback came back showing that the majority of people didn't know what the application was for when hearing the name, and when we found a formal definition for the word 'Lister' it was defined as being a farming plough, therefore we decided against this name, and decided upon 'Property Manager' which got a much better response overall and tells the user more on what the application is about.

We also designed our own logo after choosing the final name of the application. The logo shows a house outline, which is the shade of blue we have used throughout the app, with the words 'Property Manager' running along the bottom in blue and white. We also asked several people what they thought of our logo, and we were given a hugely positive response on the simple aspect of it, whilst still showing what the application is for with the use of the house, therefore we decided to use it. Although overall our designs are simple, and not particularly technical we felt that our time needed to be spent in the backend of the application to ensure all features worked as they should, as none of our group had ever created an app before. Although simple, we believe the overall design of the application works well and looks professional throughout.

5.2 Design - Website

We started the creation of the website after we were approximately half way through the creation of the application, as this was a secondary feature. As we already had the design of the application, we wanted to keep the website similar as much as possible, therefore kept to the same colour scheme and logo. Again, the website needed to be simple, as it only has minimum functionality. The main part of the website that needed a design was the Advert page. For this we looked at other adverts on Letting agents' pages such as Abode Sales and Lettings Ltd, which is the kind of thing we wanted in terms of layout (Housescape.org.uk, 2019). Our chosen layout has been added to the corpus, under the designs section. Overall, we were able to keep to this design, however a few parts were changed, such as adding the room names to the images. We also didn't include the main description in the advert, as this function was not added to the application, however this feature is easily added in the future.

6. Development

6.1 Development - Android Application

Initially we planned to develop the mobile application using the Mendix platform, but it soon became clear that although it made the user interface - UI creation very simple as it uses a drag and drop system for UI creation. It can also run code under the bonnet, but this was proving very difficult to implement and the user documentation was not easy to follow and was mostly out of date. We then decided to switch the development to Android Studio as this uses Java which we are all familiar with.

The development of the Android application was completed in weekly stages. The aim was to try to implement a new activity each week. The first stage was to implement user authentication which was first completed using the Google authentication api, shortly after I read about the Google Firebase api and decided this would be a great addition to the project as it not only handled authentication but would also provide access to Google Firestore a NoSQL database. I then rewrote the authentication class FireBaseAuthActivity to use the Firebase api which is easily setup through the Android Studio ide which imports the required dependencies into the gradle files. We had already designed the UI layout for the app which followed a clean easy to navigate design.

The next task was to create the HomeActivity the user is directed to after successful authentication, this consists of three buttons customers, properties, and logout. It also has an account button on the action bar that when opened loads the SignedInActivity which shows details about the logged in user name, email address and an image if it is a google account. The customer button would load a view that contains a list of the customers that are stored in the database and properties would display a list of the properties stored in the database. In order to create a customer database entry required a new activity that would capture the customers details.

I added a floating action button with a + symbol that would be visible over the list of customers that when touched opens the NewCustomerActivity with a form that allows the user to input the customer's name, phone number, email address and property number. I then implemented the Google Places api to create an autocomplete search field that uses the postcode to find the full address which is then stored to a hidden field on the screen. Then using the Google Firestore api a request is made for a new unique document reference number used when creating entries in the Firestore database. We then use Customer and Address Java Objects to create a Customer Object which is then saved into the Customers collection using the new reference number as a new document.

I then created the ExistingCustomersActivity which displays a list of the Customers that we have stored in the database. This was completed using a recyclerView adaptor CustomersAdaptor which handles the contents to be displayed in the activity along with the ExistingCustomersActivity. These classes work together to display a list of the customers in a card view which can be clicked to see a full-page customer details view which is also the UpdateCustomerActivity.

The next step was to implement the update functionality which is controlled by an edit button on the UpdateCustomerActivity. The default view shows all the customer details as TextView's clicking the edit button hides these views and enables hidden EditText field's that now contain the customer details enabling the user to now edit these fields. The edit button also becomes hidden and we now display a red cancel button in its place, the cancel button reverses the view back to default. Also displayed is delete and update buttons. On opening the UpdateCustomerActivity we pass into this activity a copy of the Customers Object from which we can retrieve the CustomerId. This is the unique document reference number used by Firestore to identify the document. When a customer's details are updated, and the user clicks the update button we create a new Customer Object using the data from the EditText fields. we then make a call to FireStore and search the Customers collection for the document whose reference matches the CustomerId we retrieved previously. We then call an update on that document and pass all the data from the new Customer Object. If successful, the app will display a green toasty message informing the user the update was successful. This activity was later extended to also search for any property details stored against that customer and update the customer's name on those entries when updated.

I then decided that the best way to store the property details in the database would be by linking them somehow to the customers that own them. So, I added an add property button to the UpdateCustomerActivity view that when clicked opens the CreatePropertyActivity1 view which starts the process of adding a property. Starting with the property number we then again have an autocomplete postcode search field provided by the Google Places api that allows search for an address using the postcode. The user can then select the property type, number of receptions, number of bathrooms and the number of bedrooms from dropdown menus. Clicking the next button will then start a call to the FireStore Properties collection to generate a new document reference number to use when saving the property details to the database. A new Address Object is created and added to a Property Object that contains the data of the property.

Next, I needed an activity to capture further property details. I then created CreatePropertyActivity2 which contains a toggle switch that captures whether the property is for sale or rent. This also toggles an EditText fields hint message between sale price and monthly rent. We also have switches to capture if the property has parking, a garden, double glazing and gas central heating. 1 issue I encountered here was that if you create an EditText field and set the inputType to "numberDecimal" it should allow the user to input the number 0-9 full stops and commas. However, on some devices it will show a keyboard with these values, but the comma will be greyed out and not useable. From research it looks like this bug has been present for years with no fix found. In order to combat this so we can have a price with a comma in it I have had to use full stops in place of commas when inputting a price and then add code to replace full stops with commas before storing the data in the database.

We decided that we wanted to be able to use the devices camera to capture images of the property and store them in the database with the property details. So, I decided to continue the simple flow I had started to create with CreatePropertyActivity3. This activity guides the user to use the camera to take an external property image by clicking the camera button which opens the devices camera view and after taking the picture and clicking ok brings the user back to the CreatePropertyActivity3 where the image the

user has just taken in displayed in an ImageView. The user can either press the camera button to take the image again or click the next button which will save the image. Google FireStore can only hold very small image files and they must be converted to binary form in order to do so. Instead they provide a separate storage location that can be used for images. This just consists of folder storage so in order to save the images to our properties I had to first make a call to Firebase Storage and the Property Images folder I had created and create a child folder using the properties reference number we used previously. I can then add the image to this child folder, then I needed to make a call back to this image to retrieve the download URL of the image which i can then add to the properties database entry by making a call to the FireStore database using the property reference again.

I then created CreatePropertyActivity4 here the user can input a room name from a dropdown list or click the custom button to enter a custom room name. We have EditText fields to capture the room length and width in feet and inches. In order to restrict the inches input field to a maximum of 11 I have a class InputFilterMinMax that is used to prevent 12 or higher being entered by the user. Again, we have a camera button that launches the devices camera view after capturing the image the user will see a preview of it and can click the camera button again to retake it. They would next enter a description of the room in the input field below the camera. They can either use the keyboard or the inbuilt android speech to text functionality. Finally, they either click add room which will save that room to the properties database entry and reload the activity for another room to be added or click save to complete the property creation process. Here we must again save the image to the firebase storage and then retrieve its download URL. However instead of adding the details straight to the properties document we create a sub collection called Rooms and using the Rooms Object we add the rooms details and the image URL to this sub collection this allows us to easily retrieve all the room data separately from the main property data.

The last tasks where to create a list view of all the properties saved in the FireStore database. This was easily implemented using the same process as the customer list. This required an ExistingPropertiesActivity and a PropertiesAdapter class that work together to display the properties in a list of cards. Clicking on a property would open an UpdatePropertyActivity that allows updating of the basic property details. Users can also delete the property. I then added a camera roll button to the UpdatePropertyActivity clicking this will open the PropertySlideshowActivity which uses the SlideshowAdaptor to display a list view of the rooms of the property along with the images, measurements and the description. Finally, I wanted to be able to see all properties belonging to a single customer so in the UpdateCustomerActivity I added a properties button that when clicked opens the CustomerPropertiesActivity which reuses the PropertiesAdaptor to just show properties belonging to that customer. Which can also then be clicked to update and view any room details that are stored.

6.2 Development - Website

Elly started the development of the website in week 13. To begin with we created a basic framework of the site, as we wanted to get the functionality working and focus on the adverts.html page which was the most important aspect. The first features we introduced were pulling the customer and property details in from the database. We did this by using a data call in JavaScript to get the information from Firebase and putting it into a table on the website. As Aaron had already done this for the Mobile Application, he assisted with this task. We then decided to use Materialize for the CSS (Materializecss.com, 2019). This was initially problematic in that Elly hadn't used this before and needed to learn, however we then gave Otto the task of working on the styling of the website in order to allow Elly to focus on the functionality. Once all the data was successfully being displayed on the website, we could then work on the adverts page. We started by designing on paper how we wanted the Div. elements to be laid out and created the basic frame. We were then able to pull the data in, in a similar way to how the data was pulled in on the customers.html page and the properties.html page. This was again done using JavaScript and can be seen in the adverts.js page. Overall the website development wasn't too problematic, as we had used JavaScript before and were comfortable with this. For the development of the adverts.html page Elly and Aaron conducted some pair programming in order to complete the page effectively as this was the most important page of the project, as if this did not work then the project would not have been a success.

7. Testing

The testing tables can be seen in our corpus under the testing section.
In order to test the mobile application and website, we created a list of the application and website functions and then examined them by answering the following questions:

- what is the expected outcome?
- what is the actual outcome we are getting?
- what is needed to be done?

The first area we looked at was regarding the firebase authentication logins. The mobile application supports email login and google login, whilst the website only supports the google login. The estate agent could register or use an existing email account to login, which means that the outcome satisfies our expected outcome. The mobile applications account button displays the name, email address, sign out button and a delete account button. On the other hand, the website account button popup a blank window. However, the application account button only displays this information when clicked on from the home page, which we have listed as an error in the testing table.

The application has a setting's drop menu that includes tabs for about, FAQs, and feedback options, none of them display information when they are clicked. The about option should show the current version of the application and a short description. FAQs should display some frequently asked questions and the feedback section should display a form for users to submit feedback to the applications development team.

During development we did come across a few bugs, which we were able to fix, these include:

- When we implemented email account authentication using the Firebase api on the website it initially appeared to be working. When we conducted tests on this, we discovered that in order to log in using an email and password the user would have to open the login modal twice once to select email authentication and then again in order to login. This didn't provide a good user experience, so we decided to remove email authentication and focus our time on ensuring the website worked fully. We did intend to implement this again at the end but didn't have time.
- When testing the Mobile application, a user picked up a bug with the Google Places api. When the user clicks on the autocomplete postcode search box it should open full screen with the text field gaining focus and the keyboard displaying so the user can type straight away but the text field isn't currently receiving focus and so the user must again click the search box in order to open the keyboard so they can begin typing the postcode. We have contacted the Google Places development team about this error, which they have noted as only being present in Android api 28 and they are currently tracking the issue as a bug to fix for the next release.
- We also discovered a bug in the mobile applications user deletion code that would prevent a customer with no properties assigned to them from being deleted. This was because I had coded the delete user method to search for and delete any properties assigned to a customer from the Firestore database first before deleting the customers database entry, but this meant the loop would stop if no properties where found. This was corrected by checking for properties and only deleting them if they exist.

8. Distribution

We have hosted the website for our project at: https://property-manager.pedaars.co.uk/index.html, however we have not been able to distribute the mobile application to the Android Play Store due to the cost involved. With future development of the application and website, giving it more features, we would then be able to investigate having it made available on the Playstore. At the current stage, however we do not feel it is ready.

9. Conclusions

Overall, we believe both the mobile application and Website have been a success. We have been able to fully implement all tasks that we set out to do, and they both work as intended. We have been able to identify future works, and areas in which we could improve. Both the mobile application and website have a professional appearance and share the same styling to show they are connected to each other. During this report we have discussed in detail the development process we completed and been able to reflect on this. Our idea has been original, and we cannot see that it is like any other group project this year and has been completed with the potential user in mind and the aim of distribution in the future.

9.1 Final Product

9.9.1 Final Product - Android Application

The final mobile Application has succeeded our expectations in terms of design. We have implemented all the features that we set out to complete, however due to the time constraints we were unable to complete any of our additional features that we would have liked. the features implemented are:

- Securely authenticating users logging into the application
- Saving customer and property details to a database
- Ability to edit and delete customer and property details from the database
- Implementing a simple step-by-step process for adding properties
- Using the mobile devices camera to take property photographs and then storing these in the database
- All stored data being accessible for use on the website

9.1.2 Final Product - Website

Overall, as a group we are happy with the outcome of the website. We have added all the features that we intended for this project, although there is more that can be done. Its appearance is simple, yet professional, and is fully functional. The features we have implemented are:

- Logging in via Firebase
- Viewing properties and customers in the database
- Adding properties and customers to the database
- Viewing the final advert of a property based on a given structure.
- logging out of the website

As the website wasn't the main task of our project, we didn't feel it was necessary to implement any more features, as we needed to give the mobile application more attention.

9.2 Future Work

9.2.1 Future work - Android Application

There are several features that we would have liked to have had the time to complete in the mobile application.

Creating an independent IOS version of Property Manager would expand its potential customer base. Android does have a huge share of the European market at 70.91 % compared to Apple IOS with 27.95% (StatCounter Global Stats, 2019).
Room measuring using the mobile phones camera this would allow the estate agent to easily record each rooms measurement and increase the applications functionality and usability.

During development of the application Aaron found an application called Magicplan (magicplan, 2019) that uses the device's camera to quickly create floor plans of a room. This functionality would be perfect for property manager and could either be added by

coding our own solution or using the api provided by Magicplan to integrate their functionality into our application.

9.2.2 Future work - Website

In terms of developing the website there is a lot that can be done. For the functionality of our project we wanted to keep the website simple, and solely used to aid the application, however in order to be rolled out for company use we would need to add a lot of functionality. The first area we would need to develop further is the creating a property function. You can currently only add the basic property details (address, customer it links to, number of bedrooms etc) but it will also need functionality to handle uploading of images saved on the user's computer. This would allow them to fully create properties from their desktop. We would also need to add a search bar, so the user can input either an address, or customer name to quickly pull up details. We could also allow the user to delete entries from either the customer or property database. We wanted to also display the current energy certificate for the property when generating the property listing but ran out of time to implement it this could be added using the api provided by the Ministry of Housing (Epc.opendatacommunities.org, 2019). We will also need to add further login functionality for company use. You can currently only log in via a google account login, as we ran into issues when adding email address login (it caused issues in Firebase with the applications authentication), however a company may need to use their own account already created within the application.

10. Acknowledgements

We would like to thank Colin Johnson for supervising our project, guiding us and keeping us on track. We would also like to thank Google for providing the Firebase api and for the support they provided. We would also like to thank the Android team for providing an easy and interesting operating system to work with.

11. Bibliography

Casserly, M. (2019). *Which is the more popular platform: iPhone or Android?*. [online] Macworld UK. Available at: https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/ [Accessed 22 Mar. 2019].

Epc.opendatacommunities.org. (2019). *Domestic Energy Performance Certificates API*. [online] Available at: https://epc.opendatacommunities.org/docs/api/domestic [Accessed 22 Mar. 2019].

Firebase. (2019). *Cloud Firestore | Firebase*. [online] Available at: https://firebase.google.com/docs/firestore/ [Accessed 22 Mar. 2019].

Firebase. (2019). *Firebase*. [online] Available at: https://firebase.google.com/ [Accessed 22 Mar. 2019].

Google.com. (2019). *define lister - Google Search*. [online] Available at: https://www.google.com/search?q=define+lister&oq=define+lister&aqs=chrome..69i57j0l5.2811j1j7&sourceid=chrome&ie=UTF-8 [Accessed 23 Mar. 2019].

Housescape.org.uk. (2019). *Full Details - Blackthorn Road - Canterbury*. [online] Available at: http://www.housescape.org.uk/cgi-bin/full.pl?&abd1&&ABD1000347&& [Accessed 23 Mar. 2019].

Ibisworld.co.uk. (2018). *Estate Agents (UK) - Industry Research Reports | IBISWorld*. [online] Available at: https://www.ibisworld.co.uk/industry-trends/market-research-reports/real-estate-activities/estate-agents.html [Accessed 8 Jan. 2019].
InventoryBase. (2019). *Inspection & Property Inventory Software made simple | InventoryBase*. [online] Available at: https://inventorybase.co.uk/ [Accessed 19 Mar. 2019].

Magicplan. (2019). *#1 Floor Plan App, Construction & Surveying Software | magicplan*. [online] Available at: https://www.magicplan.app/ [Accessed 22 Mar. 2019].

Materializecss.com. (2019). *Documentation - Materialize*. [online] Available at: https://materializecss.com/ [Accessed 27 Mar. 2019].

Mendix. (2019). *Application Platform As A Service - Application Development Platform*. [online] Available at: https://www.mendix.com/application-platform-as-a-service/ [Accessed 22 Mar. 2019].

StatCounter Global Stats. (2019). *Mobile Operating System Market Share Europe | StatCounter Global Stats*. [online] Available at: http://gs.statcounter.com/os-market-share/mobile/europe [Accessed 22 Mar. 2019].

Statista. (2018). *Estate agents & auctioneers 2011-2018 | UK Statistic*. [online] Available at: https://www.statista.com/statistics/319823/number-of-estate-agents-and-auctioneers-in-the-uk/ [Accessed 10 Jan. 2019].
Thesaurus.org.uk. (2019). *Cloud based Estate & Letting Agency Software from Â£26.25pm - Thesaurus Technology*. [online] Available at: https://www.thesaurus.org.uk/ [Accessed 19 Mar. 2019].

12. Appendix