# Testing Documentation

## Testing Process Followed

### Unit Testing Process

Throughout the initial prototype development, testing was used a lot to determine whether code was working properly. Unfortunately this was all manually done and very inefficient. As we had issues with getting a local server running php on a local machine, a web server was used instead. Due to this, if we wanted to test code, we would have to upload our new code to the server, refresh our browser and then test. Since then we have developed some unit tests which cover most of the functionality of our system and are using these to test it each time we make a large change. As these tests take a considerable amount of time to perform, we decided to use some scripts to save time and automate the testing process.

There are a variety of tools out there but for simplicity, and due to prior experience, we decided to use a simple scripting program called AutoHotKey. This program allows you to create hotkeys via scripts. It is very powerful and can be used to quickly perform repetitive tasks. AutoHotKey is therefore perfect for what we need, as we can just run the scripts and each test will be completed quickly and easily.

Although at least one of us has used AutoHotKey before, it had been a few years and therefore they required a quick recap on the script's syntax and how it worked. There were also some issues where scripts ran too fast in comparison to the site loading and therefore had to be slowed down. As our testing process is very much visual, we still needed someone to actually check the test results. This is a slight downfall when it comes to other automated technologies which can check the outcome of tests and tell you if it has failed or not, e.g. Java's JUnits.

### Stress Testing Process

For stress testing the site we used the site loadimpact.com. This site allows for detailed testing of websites under various conditions, such as impact of different browsers and devices. Given that we had no prior experience with loadimpact some time was needed to learn how to setup the tests and how to interpret the results. Fortunately, the site is fairly easy to use and includes a number of helpful articles for this purpose. For these tests we used a fairly generic auto generated test, with some adjustments made such as the number of users and the location of users.

Three tests were run on the site, the first simulating 25 users over 5 minutes, the second simulating 50 users over 5 minutes, and the final test simulating 100 users over 5 minutes. The results of these tests are displayed below in the stress testing section. In the first test the simulated users connected to the site from servers in the US, however for the remaining two tests this was corrected to the UK where we expect the majority of our users to actually connect from.

### Usability Testing Process

We invited some of our housemates to the library to test out our website. Initially, we were going to use www.usertesting.com. Usertesting.com is a website for customers to get on-demand feedback from a target market. But we figured we would just ask some of our friends to test and give us some feedback on the website, our website is still not that advanced therefore it would be a waste of money if we were to use usertesting.com. The usability process is discussed more in the final section where we detail our results, feedback and the step by step process we took to obtain the results.

## Unit Testing

Our current code has no actually transfer of data between functions, only between the site and the database. Each unit refers to the code related with that function, which for the site is "included" into a page when it is needed via php. We first identified the partitions of each unit so that we can reduce the number of test cases we have to do. This provides us with a good amount of pre-defined data (for testing with) of which we know what the results should be. From testing we can compare back with our partitions and see if they match, and if they don't then we will have to make changes to the code and test it again. An overview of the unit tests can be seen further below.

#### Identifying Partitions

Register

This method takes parameters; 'firstName', 'lastName', 'email', 'phone', 'username' and 'password'.

Valid intervals:

firstName, lastName, email, username, password = { string : length of 1 ≤ string ≤ length of 50 }

phone = { int : length of 8 ≤ int ≤ length 11 }; { int : 00000000 ≤ int ≤ 99999999999 }; { *null* }

Invalid intervals:

firstName, lastName, email, username, password = { string : string < length of 1 }; { string : string > length of 50 }; { *null* }

phone = { int : int < length of 8 }; { int : int > length of 11 }; { int : int < 00000000 }; { int : int > 99999999999 }

*Each tests Inputs and expected results*

| **Test** | **firstName** | **lastName** | **email** | **phone** | **username** | **password** | **Expected Results** |
|---|---|---|---|---|---|---|---|
| 1 | Matt | Hawkins | up769535@myport.ac.uk | *Null* | matt | matt | “Your Account Has Been Created.” |
| 2 | John | Smith | johnsmith@google.com | 67686797894 | john | smith | “Your Account Has Been Created.” |
| 3 | *Null* | *Null* | *Null* | *Null* | *Null* | *Null* | “Please enter some data.” |
| 4 | bob | bob | bob | bob | bob | bob | “Please enter a correct contact number.” |
| 5 | bob | bob | bob | 012345 | bob | bob | “Please enter a correct contact number.” |

## Sign in

This method takes parameters; ‘username’ and ‘password’.

Valid intervals:

username, password = { string : length of 1 ≤ string ≤ length of 50 }

Invalid intervals:

firstName, lastName, email, username, password = { string : string < length of 1 }; { string : string > length of 50 }; { *null* }

*Each tests Inputs and expected results*

| **Test** | **username** | **password** | **Expected Results** |
|---|---|---|---|
| 1 | matt | matt | “Hey there, matt.” And shows sign in options “Logout” and “Change Settings” |
| 2 | admin | admin | “Email or password entered is incorrect.” |
| 3 | 999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999 | 0 | Both fields limit characters to a max length of 50 so will reduce the input of username to the following: “99999999999999999999999999999999999999999999999999” then when sign in is clicked it should return “Email or password entered is incorrect.” |
| 4 | *Null* | *Null* | “Please enter some data” |

## Search

This method takes parameters; ‘location’, ‘type’, ‘bedrooms’, ‘minPrice’ and ‘maxPrice’.

Valid intervals:

location, type = { string : length of 1 ≤ string ≤ length of 50 }

bedrooms = { int : 1 ≤ int ≤ 10 }

minPrice, maxPrice = { decimal : 0.00 ≤ int ≤ 500.00 }; { *null* }

Invalid intervals:

location, type = { string : string < length of 1 }; { string : string > length of 50 }; { *null* }

bedrooms = { int : int < 1 }; { int : int > 10 }; { *null* }

minPrice, maxPrice = { int : int < 0.00 }; { int : int > 500.00 }

*Each tests Inputs and expected results*

| **Test** | **location** | **type** | **bedrooms** | **minPrice** | **maxPrice** | **Expected Results** |
|---|---|---|---|---|---|---|
| 1 | Portsmouth | Flat | 1 | *Null* | *Null* | Finds results with no errors. |
| 2 | *Null* | Terraced | 3 | *Null* | *Null* | “Location must be filled in” |
| 3 | 012345 | Bungalow | 5 | e | e | “Please enter a number for prices.” |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 012345 | Flat | 1 | 100 | 500 | “Sorry, we couldn't find any results.” |
| 5 | Portsmouth | Flat | 1 | -999999999999<br>9999999999999<br>9999999999999 | 999999999999<br>9999999999999<br>9999999999999 | “Please enter a number above 0 and 500 or below for your prices”<br>minPrice out of range<br>maxPrice out of range |
| 6 | Portsmouth | Flat | 1 | 100 | 500 | Finds results with no errors. |
| 7 | Portsmouth | Flat | 1 | 100 | 501 | “Please enter a number above 0 and 500 or below for your prices”<br>maxPrice out of range |

Change Settings

This method takes parameters; ‘newEmai’, ‘password’, ‘newPassword’, ‘changeSettings’ and ‘deleteAccount’.

Valid intervals:

newEmail, password, newPassword = { string : length of 1 ≤ string ≤ length of 50 }

changeSettings, deleteAccount = { bool : True }; { bool : False }

Invalid intervals:

newEmail, password, newPassword = { string : string < length of 1 }; { string : string > length of 50 }; { *null* }

*Each tests Inputs and expected results*

| **Test** | **newEmail** | **password** | **newPassword** | **changeSettings** | **deleteAccount** | **Expect Results** |
|---|---|---|---|---|---|---|
| 1 | 769535@myport.ac.uk | matt | 12345 | True | False | “Email address changed”<br>“Password changed” |
| 2 | 99999999999999999999999999999 | 0 | 999999999999999<br>9999999999999 | True | False | “Invalid email”<br>“Invalid password” |
| 3 | *null* | *null* | *null* | True | False | “Invalid email”<br>“Invalid password” |
| 4 | *null* | *null* | *null* | False | True | “Account deleted” |
| 5 | 0 | 0 | 0 | True | False | “Invalid email”<br>“Invalid password” |

# Unit Tests Summary

We used the same data for our partitions in our testing to reduce the amount of tests we needed to do, thus allowing us to be more efficient during the testing process.

*Expected results and inputs can be seen in more detail under “Identifying partitions” in the above section.*

| **Unit** | **Specification** | **Test** | **Expected Results** | **Actual Results** | **Pass/ Fail** |
|---|---|---|---|---|---|
| register | Add 4 users using normal and abnormal data input. Data used for this test is based on the “Register” partition. | Test 1 and 2 | “Your Account Has Been Created.” | “Your Account Has Been Created.” | Pass |
| | | Test 3 | “Please enter some data.” | “Please enter some data.” | Pass |
| | | Test 4 | “Please enter a correct contact number.” | “Registration Failed, Duplicate entry '0' for key 'phone'” | Fail |
| | | Test 5 | “Please enter a correct contact number.” | “Your Account Has Been Created.”<br>*6 digits for a phone number is currently allowed.* | Fail |

| | | | | | |
|---|---|---|---|---|---|
| sign in | Sign in with accounts that exist and enter numbers that should be out of bounds. | Test 1 | “Hey there, matt.” And shows sign in options “Logout” and “Change Settings” | “Hey there, matt.<br>Logout<br>Change Settings”<br>*Account “matt” exists and signs in.* | Pass |
| | | Test 2 | “Email or password entered is incorrect.” | “Email or password entered is incorrect.”<br>*Account “admin” does not exist and did not sign in.* | Pass |
| | | Test 3 | “Email or password entered is incorrect.” | “Email or password entered is incorrect.”<br>*Account “9999….” does not exist and did not sign in.* | Pass |
| | | Test 4 | “Please enter some data” | “Please enter some data” | Pass |
| search | Search with normal and abnormal data including null data (besides type and bedroom), letters in number fields, out of range numbers and numbers for location. | Test 1 and 6 | Finds results. | Finds results. | Pass |
| | | Test 2 | “Location must be filled in” | Finds results. | Fail |
| | | Test 3 | “Please enter a number for prices.” | “Please enter a number for prices.” | Pass |
| | | Test 4 | “Sorry, we couldn't find any results.” | “Sorry, we couldn't find any results.” | Pass |
| | | Test 5 | “Please enter a number above 0 and below 500 for your prices.”<br>*minPrice out of range*<br>*maxPrice out of range* | Finds results.<br>*Searching out of the range is possible, e.g. less than 0 or over 500.* | Fail |
| | | Test 7 | “Please enter a number above 0 and below 500 for your prices.”<br>*maxPrice out of range* | Finds results.<br>*Searching out of the range is possible, e.g. less than 0 or over 500.* | Fail |
| change settings | Enter normal and abnormal data, including null data and test if delete account works. | Test 1 | “Email address changed”<br>“Password changed” | Changes to database are not made. | Fail |
| | | Test 2 | “Invalid email”<br>“Invalid password” | No message displayed to user. | Fail |
| | | Test 3 | “Please enter some data” | “Please enter some data” | Pass |
| | | Test 4 | “Account deleted” | Account is not deleted. | Fail |
| | | Test 5 | “Invalid email”<br>“Invalid password” | No message displayed to user. | Fail |
| connect database | Check database connection on every page by outputting a “Connection Successful” message upon loading the page. | Test connection on each page | Every page should be able to connect to the database. | “Connection Successful” printed on all pages. Every page connects to the database correctly. | Pass |

# Component Testing

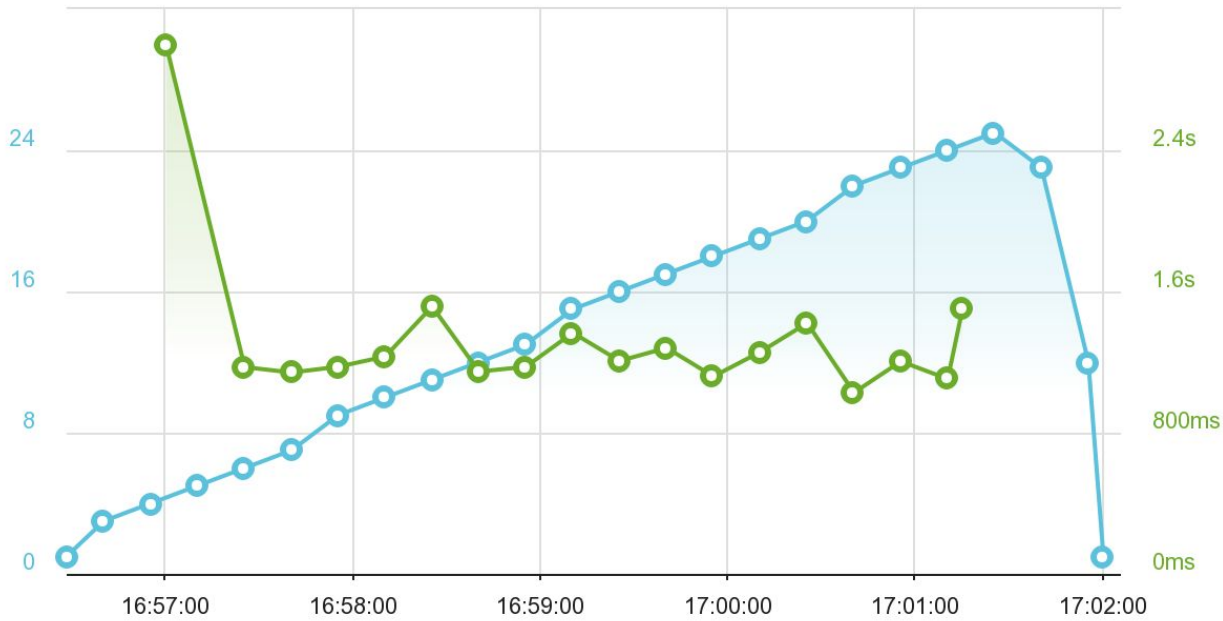
The following components and their relationship between each other and their units will need to be tested at a later date.

| Component | Purpose | Units |
|---|---|---|
| Index Page | The first page the users are greeted with, from this point the user can select which service they want to use. | connect database |
| User Component | Allows the students and landlords to create and login to their accounts in order to reach protected pages on the site. | composed off: registration, sign in, connect database |
| Search Component | Responsible for everything relating to the consultation of available listings in the database, | search, connect database |
| Management Component | The management dashboard for the landlords, from here they can add, alter or remove housing listings.This component is connected to the search component from where the database will be consulted. It is also necessary to check whether a valid user is logged in through the user component. | connect database |
| Database | Stores all the information on the website. | connect database |

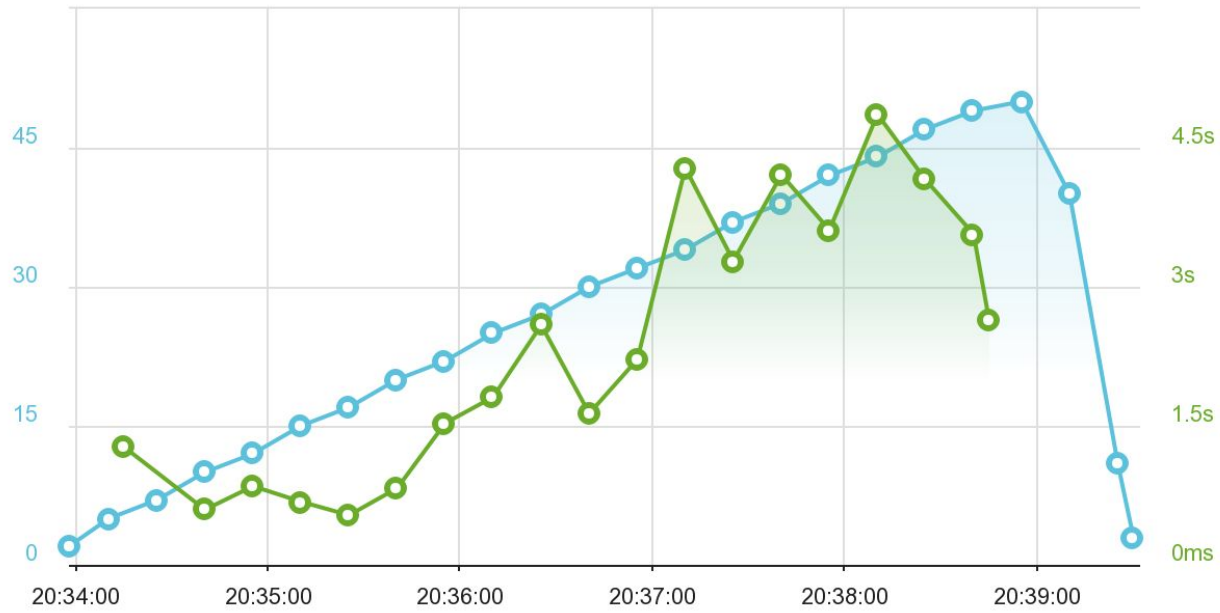
# System Stress Testing

In order to stress test the site, 3 tests were run on it. One with 25 virtual users, one with 50 VUs, and one with 100 VUs simulated. The tests produced these graphs of VU load time and number of VUs active against time, with number of VUs active in blue and VU load time in green:

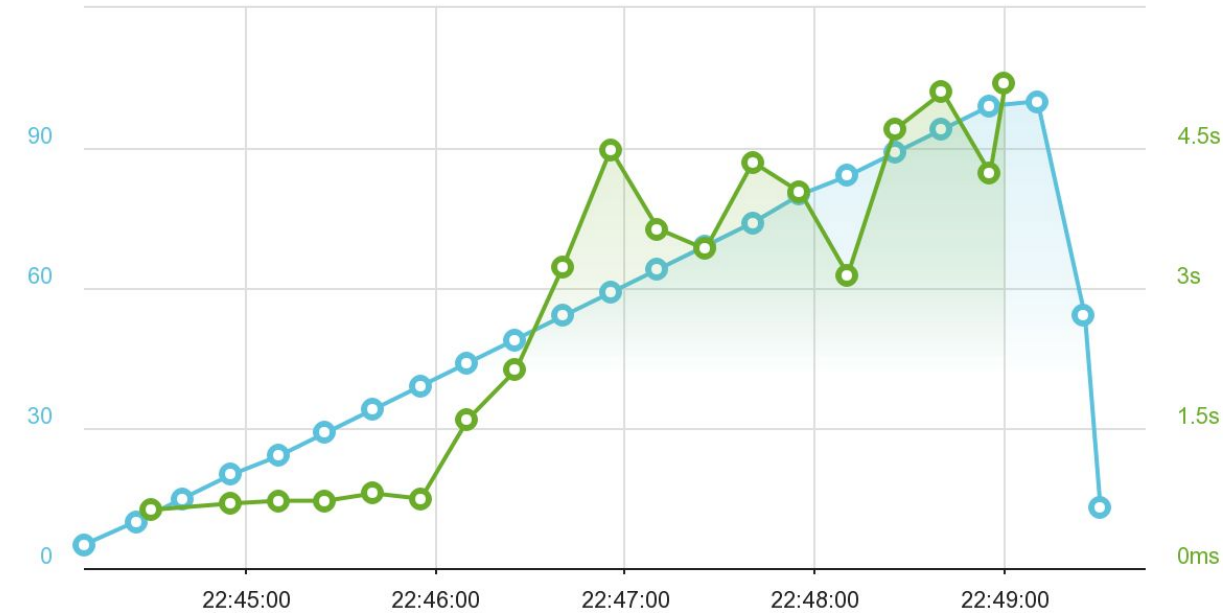
1. 25 Simulated Users



2. 50 Simulated Users:




3. 100 Simulated Users:




The tests also produced these tables which show what URLs were accessed by the virtual users along with data describing the results of the tests on these URLs.

1. 25 simultaneous users (Users were located in the US for this test but this was corrected to be in London for the following two)

| URL | Load zone<br>User scenario | Method | Status | Count | Size<br>Compressed | Min<br>Max | Avg |
|---|---|---|---|---|---|---|---|
| coursework.matthewhawkins.co | Palo Alto, US (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 143 | 1.99 KB<br>- | 380.74ms<br>5.55s | 497.63ms |
| coursework.matthewhawkins.co/../styles... | Palo Alto, US (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 143 | 1.73 KB<br>- | 335.28ms<br>1.64s | 552.1ms |
| coursework.matthewhawkins.co/../hous... | Palo Alto, US (Amazon)<br>Auto generated from coursework.matth... | Get | 404<br>Not Found | 143 | 1.33 KB<br>- | 336.35ms<br>1.44s | 539.65ms |
| coursework.matthewhawkins.co/../logo.j... | Palo Alto, US (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 143 | 24.7 KB<br>- | 502.39ms<br>1.71s | 725.29ms |
| coursework.matthewhawkins.co/../script... | Palo Alto, US (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 143 | 285 Bytes<br>- | 169.32ms<br>909.09ms | 189.59ms |

The first test showed that the site can cope with 25 simultaneous users with no issues. The VU load time was kept consistently low (aside from an initial spike) and the URL results showed that no pages were unreachable due to the traffic load. The average time to access the pages was also low on average *see graph 1*.

2. 50 simultaneous users

| URL | Load zone<br>User scenario | Method | Status | Count | Size<br>Compressed | Min<br>Max | Avg |
|---|---|---|---|---|---|---|---|
| coursework.matthewhawkins.co | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 266 | 1.99 KB<br>- | 79.56ms<br>5.28s | 237.32ms |
| coursework.matthewhawkins.co/../styles... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 212 | 1.73 KB<br>- | 36.33ms<br>5.04s | 2s |
| coursework.matthewhawkins.co/../styles... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 54 | 489 Bytes<br>- | 33.59ms<br>59.6ms | 35.41ms |
| coursework.matthewhawkins.co/../hous... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 404<br>Not Found | 190 | 1.33 KB<br>- | 36.77ms<br>5.04s | 1.87s |
| coursework.matthewhawkins.co/../hous... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 76 | 489 Bytes<br>- | 33.71ms<br>39.84ms | 34.47ms |
| coursework.matthewhawkins.co/../logo.j... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 250 | 24.7 KB<br>- | 53.57ms<br>5.05s | 2.01s |
| coursework.matthewhawkins.co/../logo.j... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 16 | 489 Bytes<br>- | 33.87ms<br>36.15ms | 34.54ms |
| coursework.matthewhawkins.co/../script... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 266 | 285 Bytes<br>- | 19.4ms<br>1.55s | 186.29ms |

The second test showed that the website encountered a few issues under the load of 50 concurrent users. The VU load time was higher overall as more virtual users accessed the site, and the URL results show that some pages on the website were unreachable due to the load and displayed 503 Service Unavailable errors. The average time to reach the pages also increased in this scenario *see graph 2*.

3. 100 simultaneous users

| URL | Load zone<br>User scenario | Method | Status | Count | Size<br>Compressed | Min<br>Max | Avg |
|---|---|---|---|---|---|---|---|
| coursework.matthewhawkins.co | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 506 | 1.99 KB<br>- | 81.94ms<br>6s | 841.24ms |
| coursework.matthewhawkins.co | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 15 | 489 Bytes<br>- | 36.37ms<br>39.71ms | 37.3ms |
| coursework.matthewhawkins.co/../styles... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 290 | 1.73 KB<br>- | 36.11ms<br>5.04s | 2.35s |
| coursework.matthewhawkins.co/../styles... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 224 | 489 Bytes<br>- | 36.17ms<br>64.87ms | 37.63ms |
| coursework.matthewhawkins.co/../hous... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 404<br>Not Found | 261 | 1.33 KB<br>- | 39.28ms<br>5.01s | 2.17s |
| coursework.matthewhawkins.co/../hous... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 253 | 489 Bytes<br>- | 36.2ms<br>64.22ms | 37.54ms |
| coursework.matthewhawkins.co/../logo.j... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 391 | 24.7 KB<br>- | 53.69ms<br>5.06s | 2.56s |
| coursework.matthewhawkins.co/../logo.j... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 123 | 489 Bytes<br>- | 36.4ms<br>53.13ms | 37.4ms |
| coursework.matthewhawkins.co/../script... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 200<br>OK | 494 | 285 Bytes<br>- | 19.84ms<br>4.93s | 471.78ms |
| coursework.matthewhawkins.co/../script... | London, GB (Amazon)<br>Auto generated from coursework.matth... | Get | 503<br>Service<br>Unavailable | 20 | 489 Bytes<br>- | 18.25ms<br>21.12ms | 19ms |

The third test demonstrated that the site encountered more issues with 100 concurrent virtual users. Again the VU load time increased along with the number of virtual users, however it did not increase by much compared to the 50 users test, with both reaching a maximum of just over 4.5 seconds. There were also more 503 errors encountered by the virtual users, and the average time to reach the pages increased considerably *see graph 3*.

# Results of the Testing Process

The unit testing made it clear that we have some issues left in the system that need fixing. After finding these issues we made changes to the system. Once the next big set of changes are made then more testing will be conducted to ensure that these unit test results still pass.

## Changes made due to unit testing

### Changes to unit “register”

- Fixed database error which caused the error in Test 4.

- Added correct validation so that users can't add a phone number that has less than 7 digits.

**Changes to unit "sign in"**

All tests passed therefore no changes required.

**Changes to unit "search"**

- Added validation so that users must enter a location.
- Added validation to prevent numbers below 0 and above 500 being input into minPrice and maxPrice.

**Changes to unit "change settings"**

- Fixed an error that prevented users details from changing on the database.
- Added validation for invalid emails and passwords

**Changes to unit "connect database"**

All tests passed therefore no changes required.

### Changes made due to stress testing

After the stress testing showed that the website suffered under many concurrent users, we thought that we should figure out a way to account for this much traffic. One way of improving this would be to migrate the site to a server better equipped for multiple users. However, the site is not expected to undergo very high traffic so this may be unnecessary.

### Changes made due to usability testing

After the usability testing, we have changed the font used on the website so that it will be easier for our users to read. Other than that, we have almost completed adding information onto the "about" page of the website. We are still figuring out how to make a working "forgot password" button, this will surely be added to our website in the future.

## Usability Evaluation Process and Results

Usability is the quality attribute that assess how easy user interface are to use. As a group we observed 5 people who had the opportunity to use our application to enable us to asses the usability of the application that we've developed. Through the analysis and feedback we received from the process we intend to use every aspect of feedback to make our application more user friendly.

### Process Taken During Usability Tests

By taking into consideration heuristic guidelines such as Nielsen's 10 Usability Heuristics for User Interface Design, we attempted to design our system while taking usability into consideration. To fully test and make sure that we had succeeded in this, we gathered 5 people to test our system.

We implemented a simple but effective method for measuring usability, the system was tested with sample users and results were recorded. During the testing the users where given set tasks and the time taken to complete these tasks was measured. During this process, the users simply did each task while telling the examiner what they were doing and what they were thinking during the process of completing each task. This allowed us to clearly see where the gaps are in our system and where users are likely to go wrong.

After this process was completed, we analysed the results of the tests. We then began a new version of the system that implemented the improvements needed. In the future we will perform this testing again to increase the usability of the system.

### Process Taken to Calculate the System's "Usability"

In Usability Metrics (2001, para 5) Nielsen lists the most basic measures of usability which he lists as follows:

- Success rate (can users perform the task?)
- Time taken to complete a task
- Error rate
- User satisfaction

He also mentions other possible metrics such as the amount of times users need to backtrack to find the correct window/page. He then goes on to discuss comparing two designs and how to quickly tell if a new design has improved in usability. He recommends taking tasks that the user must do then to record how long it takes the user to complete the task.

To do this you would gather the data for each design first then, for each one, add up the time taken so you get a number which he calls "how long it takes users to do stuff" for each design. From here you can simply calculate a

percentage improvement based on the time difference. He also means though that this can be misleading as if certain tasks are not performed as often but improve in efficiency then these tasks could skew the results. When some tasks are performed more than others, it is better to work out percentage of improvement separately first and then get a geometric mean of the tasks percentages. By doing this you then get a fair “usability score” which can tell you just how higher or lower usability is for a new design. By doing this you can get an idea as to how usable the system is and then compare it with other versions or prototypes.

As this was a relatively simple way of figuring out how usable the system was we decided to do this with task performed by the user. We can then compare this with future implementations of the system to see if we have improved our “usability”.

## Usability Task Test Results

| Person | Task | Time taken | Feedback |
|---|---|---|---|
| person1 | Registering an account | 41 seconds | Straightforward process |
| | Signing in | 25 seconds | Pretty easy task to do, found it quite difficult to read the writing due to small font. |
| | Search for accomodation | 6 seconds | |
| | Change email address | 16 seconds | |
| | Getting landlord contact info | 12 seconds | Achieved by a simple click on a button |
| person2 | Registering an account | 33 seconds | Easy to do, but there’s no option to double check my password |
| | Signing in | 19 seconds | |
| | Search for accomodation | 10 seconds | No pictures |
| | Change email address | 15 seconds | |
| | Getting landlord contact info | 12 seconds | |
| person3 | Registering an account | 38 seconds | |
| | Signing in | 18 seconds | |
| | Search for accomodation | 8 seconds | Needs some pictures of the flat/rooms |
| | Change email address | 10 seconds | A password confirmation would provide more security |
| | Getting landlord contact info | 20 seconds | Purpose of the button should be made more clear |
| person4 | Registering an account | 40 seconds | |
| | Signing in | 12 seconds | |
| | Search for accomodation | 10 seconds | |
| | Change email address | 8 seconds | A message confirming the change of my email would be nice |
| | Getting landlord contact info | 16 seconds | Could just put it on the house renting page |
| | Registering an account | 29 seconds | |

| | | | |
|---|---|---|---|
| person 5 | Signing in | 15 seconds | should include a button for when someone forgets their password |
| | Search for accomodation | 10 seconds | pictures would be nice |
| | Change email address | 11 seconds | |
| | Getting landlord contact info | 9 seconds | |

## Feedback from Usability Tests

One of the observations that we made when the users were using the application was that they were able to complete the most basic task without any issues with the system. But on the other hand when the user was first using the application there was a lot of uncompleted sections such as the “about” page and the “advertise” page on the webpage which means they weren't able to use the application to its full potential.

Another observation we made of a user using the application is that when they were trying to enter their login details into the account they spent a lot of time on the login page due to the fact that there wasn't any guidance to enable them to get better understanding of how to login into the system. The lack of guidance within the application doesn’t make for a very user friendly app which means as the developers of the application we will have to ensure that the finished product meets a more user friendly standard.

Within the basic understanding of usability we all understood that learnability is one of the most important factors of the application. Learnability determines how easy it is for a user to accomplish basic tasks the first time they use the application. On the section of learnability, the feedback we received from the participants was that the application operates on the basic foundation of most websites, where every link incorporated into the app works as expected.

One of the people that we interviewed mentioned that the login section on the homepage was quite hard to log into due to the fact that the font is very small and unreadable for some. this could cause a lot of problems with people with a bad range of view and people that may need glasses will find it very difficult to read. a way to improve this is by simply making the font slightly bigger so that the general public is able to read it.

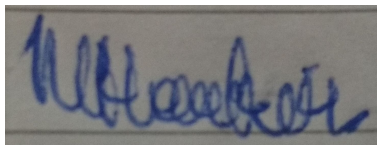
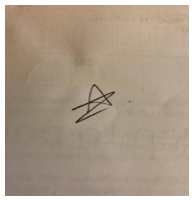
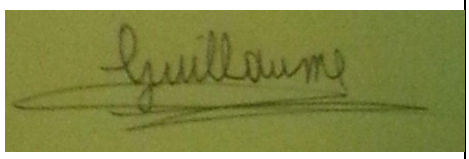
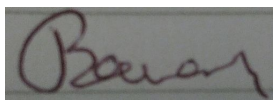
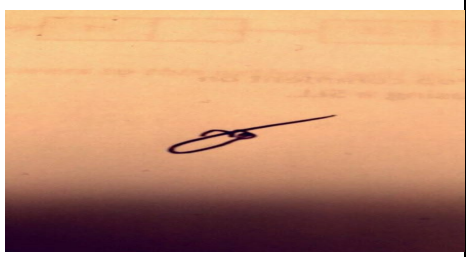
In terms of satisfaction with usability, during the evaluation of the 5 users we came to the understanding that the users weren’t satisfied with the final outcome of the application. Which means from the evaluation, we as the developers, will have to better improve the application to meet a good usability standard for our future users.

Another observation that was made is that there is no option for the user if he/she was to forget their password. This could cause the user big problems if, for example, they have a house viewing and go back to check the address of the house but are unable to, due to forgetting their password. A way to fix this is to add a “forgot password” option that allows the user to reset their password using their email.

### Works Cited

Nielsen, J. (2001). *Usability Metrics*. Retrieved from https://www.nngroup.com/articles/usability-metrics/

## Declaration

| Name | Has worked on | Signature |
|---|---|---|
| Matthew Anthony James Hawkins | “Testing Process Followed”, “Unit Testing”, “Results of the Testing Process (Unit Testing)” and “Usability Evaluation Process” |  |
| Edward Bonsu | Usability Evaluation Process | |
| Hong Hon Ping | Usability results |  |
| Guillaume Van de Sype | Description of component testing, usability testing results |  |
| Joshua Bowen | Stress testing |  |
| Mouloud Diram | Usability Evaluation Process/ results of changes due to usability and testing. |  |