
Testing Policy - EstateMatch

TeamBlue



Team contact: teambue.cos301@gmail.com

1. Performance Testing:

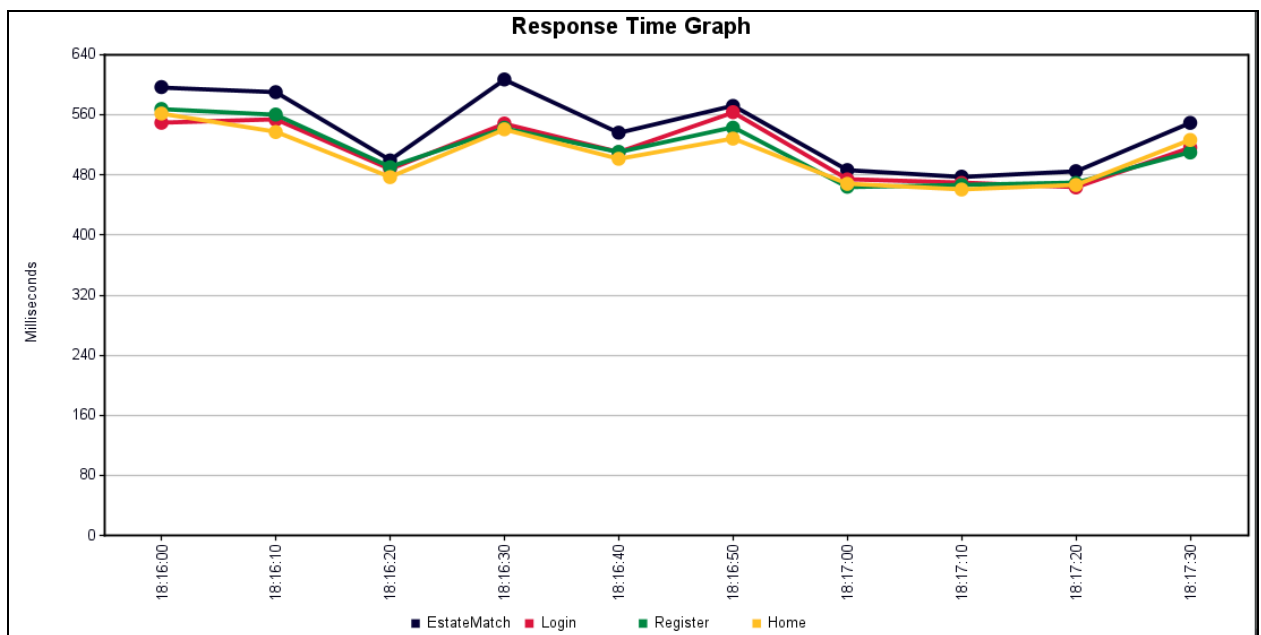
2 Types of testing techniques were used for Performance Testing

- Load Testing
- Stress Testing

Load Testing is a type of testing that evaluates how your application performs under expected load conditions. It involves simulating real-world user traffic and measuring response times, throughput, and resource utilisation. Load testing helps identify performance bottlenecks and helps you determine if your application can handle the expected user load.

For Load Testing, Apache JMeter (<https://jmeter.apache.org/>) was used to help discover the response of the following pages:

- EstateMatch (when the app is loaded)
- Login Page
- Register Page
- Home Page (where the user is able to swipe property)



Testing Property:

1000 users were set to load on each page simultaneously at a ramp-up period of 100 seconds. This means that JMeter will start 100 users every second until all 1000 users are running on the app. This helps stimulate a gradual increase in the load on the server

X-axis : Time test was run

Y-axis : Milliseconds for the Loading time

Upon further investigations, the limits of how many users can load, 7364 users will be able to load onto to the 4 pages simultaneously then the server will bottleneck

Stress Testing is a type of testing that involves pushing the system beyond its normal operational limits to identify the breaking point

For Stress Testing, Apache JMeter (<https://jmeter.apache.org/>) was used to help discover the breaking point of the following pages:

- EstateMatch (when the app is loaded)
- Login Page
- Register Page
- Home Page (where the user is able to swipe property)

A Response Assertion was used to find if each page had the correct response code (code 200), this allows us to know how many users were able to enter the application before it breaks.

Testing Property:

10 000 users were set to run twice on each page to find the breaking point of each page. Upon running the test, it was found that the breaking point for each page is on average 8542 users. (This result was found in a way where JMeter had to restart due to the amount of users tested)

2. Availability Testing:

Availability is the degree to which an application is operational and accessible when it is needed by users. It is a measure of the system's readiness to perform its intended functions without interruption.

Availability is a crucial part to any application, but more so for Estate Match due to the time sensitive nature of buying properties. We evaluated Estate Matches availability using two formulas:

$$\text{Availability: } \frac{\text{Total Uptime}}{\text{Total Time}} \times 100$$

$$\text{Error Rate: } \frac{\text{Total Errors}}{\text{Number of Requests}} \times 100$$

Availability is expressed by taking in the total time and its uptime. We take the uptime and divide it by total time and then multiplying by a 100 to give us availability as a percentage.

The Error Rate is expressed by taking the number of requests to the application that resulted in errors and the total amount of requests made to the application and dividing the two, it is then multiplied by 100 to give the error rate as a percentage.

We aim to have the availability above 95% and the error rate under 5%.

Here are readings taken from 24th August 2023 to 25th September 2023. We collected the readings by pinging the application at intervals to determine its availability and sending GET requests at intervals to the application to determine its error rate.

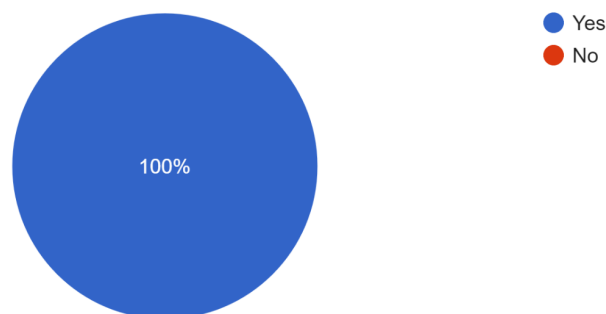
3. Usability Testing:

To test the usability of our app, we had 15 people use our app as potential clients and afterwards asked them to complete a survey about their experience using EstateMatch. The questions in the survey the user was asked to complete were mostly aimed at whether the user had a satisfactory experience using our app, and whether they thought our app was user-friendly, easy-to-use, intuitive and efficient. The questions asked during the survey were:

1. Do you think that the layout of the pages is clear and easy to navigate?
2. Are you likely to use EstateMatch again if you are ever looking for another property?
3. Do you agree that the UI (user interface) of the app is user-friendly?
4. On a scale of 1-5 (1 being unlikely and 5 being very likely) how likely are you to recommend EstateMatch to a friend?
5. Would you say that it is easy to navigate to your required page on the EstateMatch app?
6. Do you think that the font size and style is clear and easy to read?
7. Were you able to easily understand how to use the app?
8. Did you find that the app was easier to navigate due to the option to translate it to your preferred language?

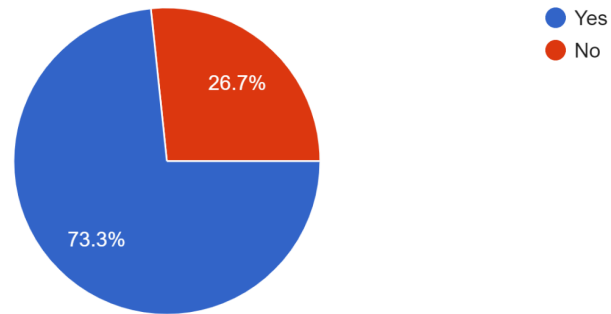
Do you think that the layout of the pages is clear and easy to navigate?

15 responses



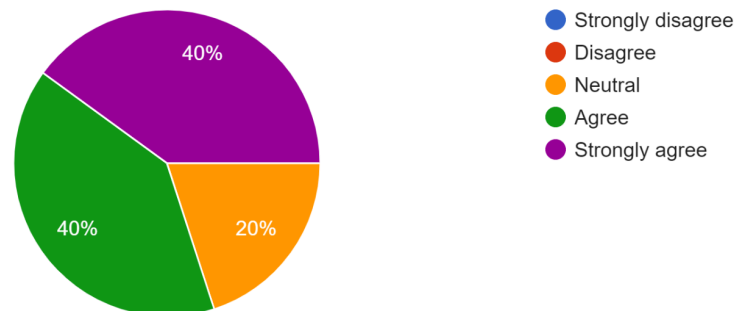
Are you likely to use EstateMatch again if you are ever looking for another property?

15 responses



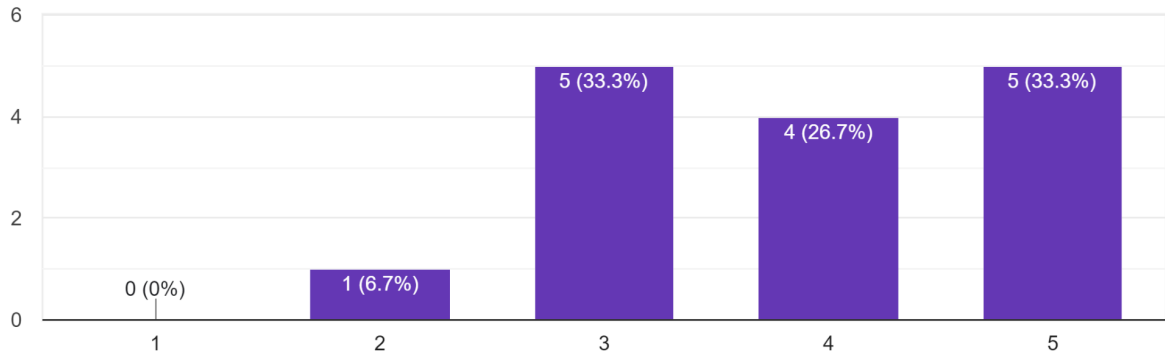
Do you agree that the UI (user interface) of the app is user friendly?

15 responses



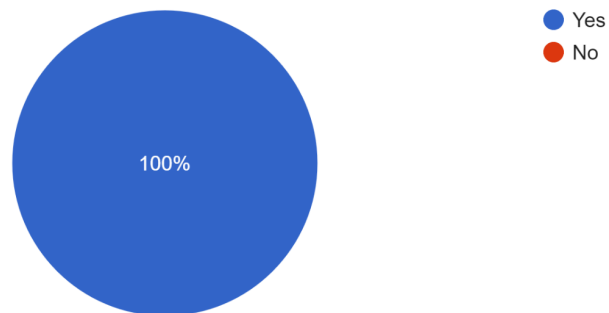
On a scale of 1-5 (1 being unlikely and 5 being very likely) how likely are you to recommend EstateMatch to a friend?

15 responses



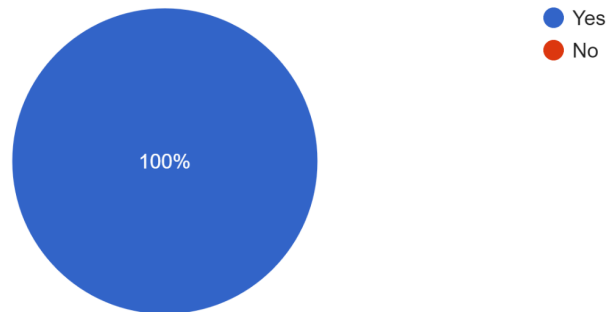
Would you say that it is easy to navigate to your required page on the EstateMatch app?

15 responses



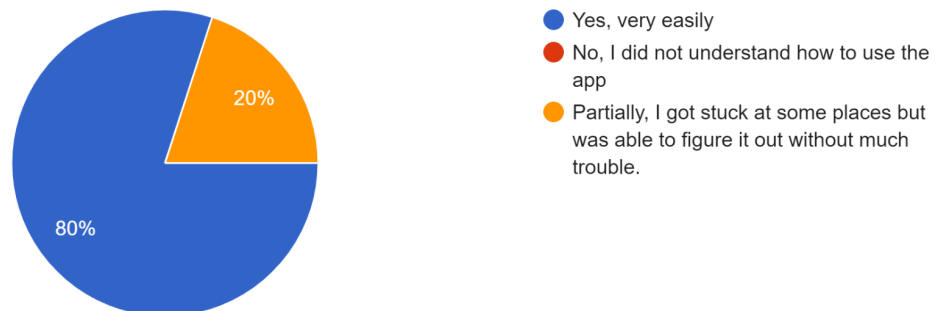
Do you think that the font size and style is clear and easy to read?

15 responses



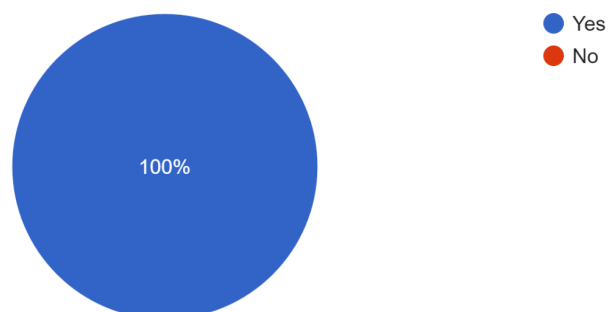
Were you able to easily understand how to use the app?

15 responses



Did you find that the app was easier to navigate due to the option to translate it to your preferred language?

15 responses



The overall response from users was that EstateMatch that using the app to look for houses was a satisfactory experience, and that they would use the app again in the future, as well as that they would recommend the app to a friend.

Therefore, we believe we have met our usability quality requirement.


4. Maintainability Testing:

The checklist for maintainability testing is the following:

1. Verifying the development standards such as structured programming, standards for database approach, recognizable nomenclature and standards for the user interfaces
2. Verify if the data processing is split up into subtransactions?
3. Verify if the input, the processing and the output have been implemented separately
4. Verify if the programs have been parameterized under necessary conditions to promote reusability.
5. Verify if the systems are distributed.
6. Verify if the algorithms are optimised.

From the start of the project we developed the system with maintainability in mind, therefore we ticked off every point of the checklist by doing the following:

1. All the members adhered to the same coding standards and all interfaces were standardised with regards to fonts, text size and colours.
2. Data processing was split up into the following subsections:
 - a. Queries and Command Handlers (CQRS).
 - b. Repositories where all the queries to the database are.
3. All the input was implemented as request interfaces, processing as commands and queries and the output as response interfaces.
4. The end-points have been parameterized by the request interfaces.
5. Systems are distributed as follows:
 - a. Model (User Interfaces)

- 
- b. Controllers and services
 - c. Command Handlers
 - d. Query Handlers
 - e. Repositories
6. Based on the results of the performance testing, we believe that our algorithms are optimal.

It is clear that we did meet the requirements for a maintainable system, therefore our system is maintainable