# Section 4 – Evaluation

## Part (A) – Post Development Testing

As outlined in the design section of my project, I originally planned on using four method of testing:

* White box testing
* Black box testing
* Destructive testing
* Unit testing

However as my project progressed I realised how long it was going to take to finish it. I then chose to ignore unit testing as it was very time intensive and didn’t provide many benefits for a project this small scale (compared to a project made and maintained by a team of people over several years). This is bad in terms of maintainability, so if I was continuing to develop this project, writing unit tests would be one of my first priorities to ease future development. I did use all three of the other testing approaches.

I also performed tests that didn’t require any user data such as simulated load tests on the server and database. This is useful to see how the website would perform if it had lots of concurrent users.

### White box tests

White box testing is when a programmer with access to the source code tests the program for functionality and to locate bugs. They look at the source code and test legal and illegal inputs, as well as checking every section actually does what its supposed to do. I used white box testing in my project as my primary testing method because it’s quick and it doesn’t require asking anyone else to test it, which is good for pre-release testing. I tested all the scripts that required input using white box testing by checking to code to see what could trip the program up, and then testing it against expected results. The results of my white box testing are below:

Browser tests:

|  |  |  |
| --- | --- | --- |
| **Browser** | **Works/Doesn’t work** | **Notes** |
| Mozilla Firefox | Works | None |
| Google Chrome | Works | None |
| Internet Explorer 11 | Works | None |
| Microsoft Edge | Unknown | Won’t connect to website at all so impossible to test |
| Mozilla Firefox (Android) | Mostly works | Site hasn’t been specifically designed for mobile but most pages work. |
| Google Chrome (Android) | Mostly works | Site hasn’t been specifically designed for mobile but most pages work |

Basic functionality:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page** | **Test** | **Test data** | **Works/Doesn’t work** | **Notes** |
| signup.php | Can create an account | All valid data | Works | Script creates an account using the data provided |
| signup.php | Can create an account | No data | Works | Script prompts user to fill in missing fields |
| signup.php | Can create an account | ☺ ☺ ☺ ☺ ☺ | Works | Script prompts the user to try again |
| login.php | Can log in to existing account | Username and password for an account existing in the database | Works | Script logs user into account |
| login.php | Can’t log into account with wrong password | Username for an account that exists in the database, wrong password for said account | Works | Script prompts user that username/password combination is invalid |
| login.php | Can’t log into account that doesn’t exist | Username and password that doesn’t exist in database | Works | Script prompts user that username/password combination is invalid |
| ask.php | Can ask a question with valid inputs | Valid question title and question body | Works | Script adds a new question to the database |
| ask.php | Can’t ask a question with nothing filled in | No data | Works | Script prompts the user to fill in every field |
| ask.php | Can ask a question containing unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script adds a new question to the database |
| searchq.php | Can search for a question with valid inputs | Valid search String | Works | Script shows all questions that match the search query |
| searchq.php | Can’t search for a question without supplying any query | No data | Works | Script prompts the user to search for questions |
| searchq.php | Can search for a question containing unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script shows all questions that match the search query |
| comment.php | Can comment on a question using valid input | Valid input comprised on standard characters | Works | Script adds a new comment to the database |
| comment.php | Can’t comment on a question without supplying a comment | No data | Works | Script prompts the user to input a comment |
| comment.php | Can comment on a question using unexpected input | ☺ ☺ ☺ ☺ ☺ | Works | Script adds a new comment to the database |
| findpeople.php | Can post a job listing using valid data | Valid job title, description, location, company, salary and tags | Works | Script adds a new job to the database |
| findpeople.php | Can’t post a job listing with no data | No data | Works | Script prompts the user to fill in all the fields |
| findpeople.php | Can post a job listing with unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script adds a new job to the database |
| findpeople.php | Can add a valid tag | A valid tag | Works | Script adds a new tag to the filter |
| findpeople.php | Can’t add a tag with no data | No data | Works | Button to add tag does not appear if the input box is empty |
| findpeople.php | Can add a tag with unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script adds a new tag to the filtered |
| findPeople.php | Can search for a job with valid input | Valid search String | Works | Script shows all jobs that match the search query |
| findPeople.php | Can’t search for a jobs without supplying any query | No data | Works | Script prompts the user to search for jobs |
| findPeople.php | Can search for a job containing unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script shows all jobs that match the search query |
| findjobs.php | Can add a valid tag | A valid tag | Works | Script adds a new tag to the filter |
| findjobs.php | Can’t add a tag with no data | No data | Works | Button to add tag does not appear if the input box is empty |
| findjobs.php | Can add a tag with unexpected characters | ☺ ☺ ☺ ☺ ☺ | Works | Script adds a new tag to the filtered |

### Black box tests

Black box testing is when the tester has no knowledge of the internals of a program. They do not have access to the code and they don’t know how it works. Black box tests are often used to test functionality and interfaces. I’m using black box tests because the final version of project is so large and complex that it’s very difficult for me to test every single feature. I also use it because I’m biased in favour of the design and interface, and black box testing allows me to see lots of user’s opinions. This will allow me to see if it meets my success criteria.

In order to black box test my project I created a survey using Microsoft Forms. The survey has seven questions designed to test various aspects of my site. When I was designing the survey I tried to make questions that test the parts of the site that I would struggle to do myself, such as opinions on the user interface and how easy it is to use the site etc.

The testers who will be black box testing my project are a class of 14-15 year old GCSE Computer Science students. This is because they are available for me to use for testing the project. This is a very effective way of me getting 30 people’s opinions on my website.

The list of questions in my survey and a link to it are below:

1. Did you like the design of the website?

(yes or no, compulsory)

2. Did you struggle to do anything on the website?

(yes or no, compulsory)

3. Did you find the tutorials easy to follow?

(yes or no, compulsory)

4. Is the interface memorable? (Would you remember how to use the website after a break of a few months?)

(yes or no, compulsory)

5. Was it easy to use each of the features for the first time?

(yes or no, compulsory)

6. What (if any) improvements would you like to see?

(text input, optional)

7. Please list any bugs you saw along with steps to reproduce them

(text input, optional)

Link to survey:

<https://forms.office.com/Pages/ResponsePage.aspx?id=PYCwG3TptkKUJQEK3ByuatBasbOGJYRIgymUpAILxi1UOFYyUVZDV05SVkJSTVdLU0JDQ0RZMFVCWS4u>

The responses I got from my survey were very positive. In total 21 people responded, which I consider a big enough sample size to draw meaningful conclusions. On average people took exactly three minutes to complete the survey which shows people thought about their answers. This makes me more confident in the data. Analysis of the survey responses are below:

1. Did you like the design of the website?

86% (18/21) of people said they liked the design of the website. This means that the user interface and design of my website was successful. As a programmer I’ve always struggled with design and making things look nice, and I was please at the feedback in this category. The design of the website is the first thing a user sees when they first use it, and having something that people find visually pleasing is essential for keeping people using the site.

2. Did you struggle to do anything on the website?

71% (15/21) of people said they didn’t struggle to do anything on the website. While this is slightly lower than the design feedback, it is still very positive considering the majority of the testers were not the target audience for my project. During my analysis I predicted that most of my users would be amateur and more experienced programmers, mostly employed/looking for work, however the testers are all 14-15 and just starting to study computer science. This means that figure of 71% is even better, which shows that some of the design work I did paid off. If people struggle to use a site they’ll give up using something straight away.

3. Did you find the tutorials easy to follow?

86% (18/21) of people agreed that the tutorials I wrote were easy to follow. Unlike the some of the other features in my project, the tutorial pages are aimed directly at the kind of people who were testing my project. I was happy with the high percentage of people that thought the tutorials were easy to follow because it shows that they worked as tutorials. There’s no point having hard to follow tutorials as they're frustrating and don’t end up helping people.

4. Is the interface memorable? (Would you remember how to use the website after a break of a few months?)

71% (15/21) of the testers considered the interface memorable enough to be able to use the website after a break of a few months). Memorability was one of my success criteria, and I consider 71% to be enough to pass this criteria. Memorability is a hard aspect to test for, because it really needs the testers to take a break of a few months, however there wasn’t enough time for this. Because of this I’m not surprised that the same percentage of people that didn’t struggle to do anything on the website also considered the interface memorable. I included this in the survey because I wanted to get people’s initial impressions, and I was pleased with the feedback from this question.

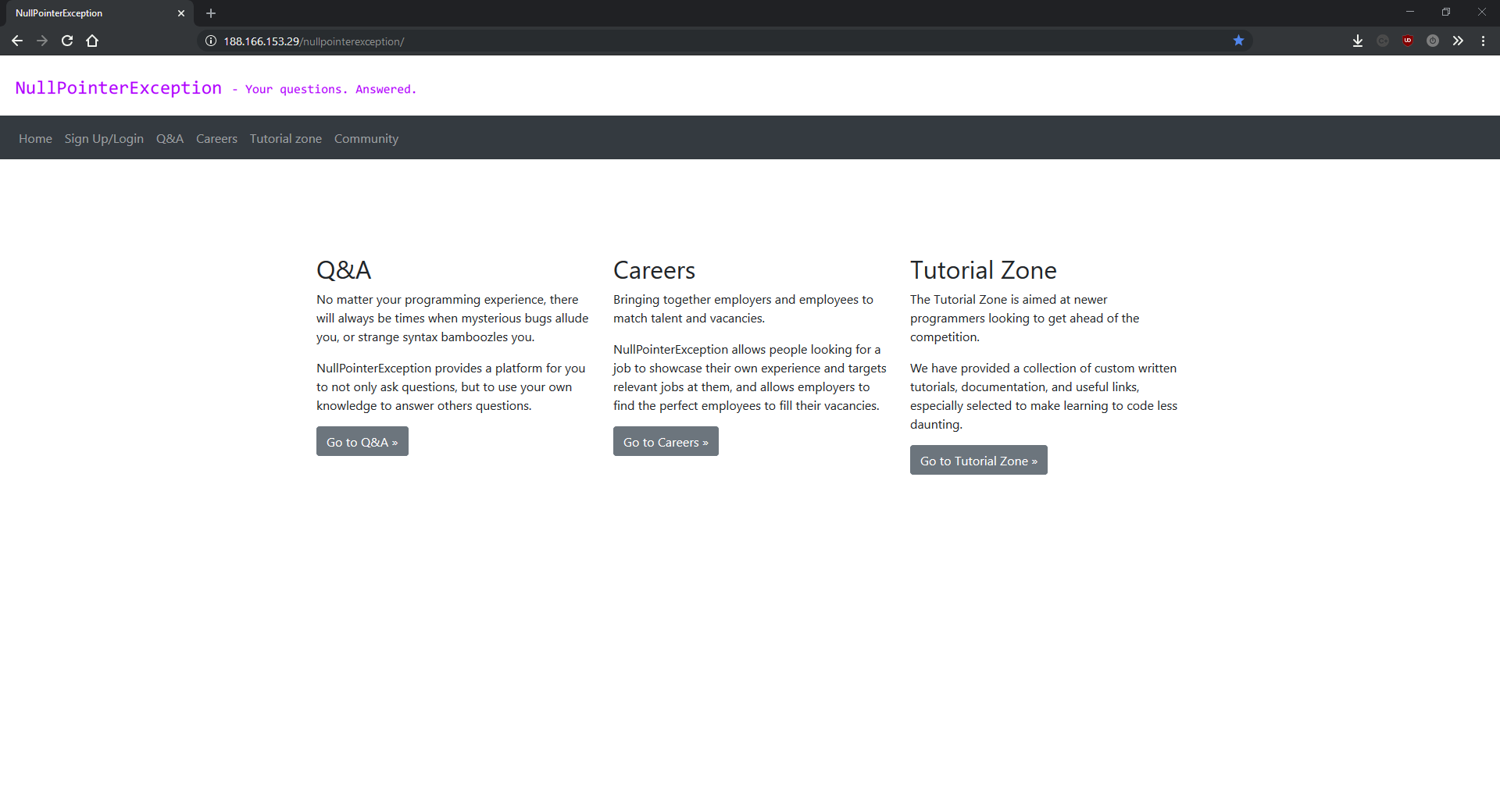
5. Was it easy to use each of the features for the first time?

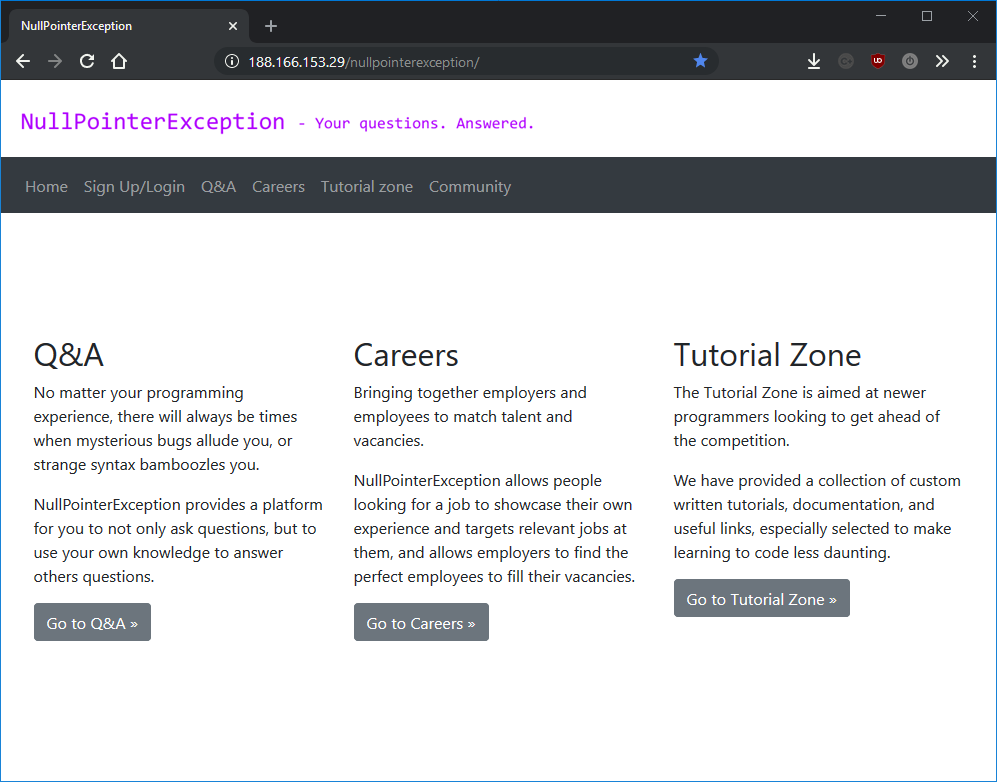
This is a harder question to analyse because I added a third answer to this question. 67% (14/21) of people thought every feature was easy to use for the first time and a further 29% (6/21) thought most of the features were easy to use for the first time. Only 1 person said that none of the features were easy to use for the first time. This is quite a positive response, but also gives me some insight one what to work on in the future.

6. What (if any) improvements would you like to see?

The final two questions in the survey were both optional and aimed at things for the future rather opinions about the current build. This question received seven responses. Two of theses were about making the layout more interesting and take up more space on the screen. While I see what they’re on about, the layout is designed to work well on tablets and phones as well as large computer screens, which means the larger screens end up with more free space. For example, the gap between sections of content and the edges of the screen will be larger on bigger screen resolutions. This can be seen below:

Full size homepage (original image 1920x1080):

Small window:



You can see that the gap between the edges of the screen expands on bigger screen sizes. This is because an easy way of creating a responsive website is to design for small screens and expand outwards. This is because a website designed for screen will work on that size and anything bigger (when done properly), but a website designed for a large 4k screen won’t work on a small phone. In response to this feedback I would redesign the website on larger screens, as well as creating a separate version for phones, all of which should help address people’s criticism.

Another response I got from this question was someone who wanted the tutorials to be broken up a bit and instead of just big paragraphs of text. I completely agree with this point, if you look at any popular tutorials on the internet they are a step by step guide to making a finished project, with the final code shown at the end. This is a better format for tutorials as having big blocks of text can be daunting for some users, and is less interesting. The reason I chose to do this approach is because the tutorials on the site at the moment are more like proof of concepts than finished articles. This is due to the limited time I had to develop this project. In the future I would like to redesign the existing tutorials as well as making more tutorials for other popular languages such as Java/C++.

One tester asked for more colours and a logo. I actually have a logo but since it’s made of text some people might not have realised what it was. In the future I might look at creating a more unique design, possibly with a bit more colour, but for now I’m happy with the current design, thanks to the feedback from question one.

7. Please list any bugs you saw along with steps to reproduce them

This was a very useful question to ask. While the other questions have been testing for usability and opinions on design, this question is testing for functionality and robustness. I was expecting either of two options when I asked this question, either the testers would find no bugs meaning that the website is fully functional and works perfectly, or the testers would find bugs that I’d missed during development along with instructions as to how to recreate them. Three users discovered a bug which I hadn’t considered, even though it’s something I was meaning to address since day one. The bug they found was when you used apostrophes in any of the input field that gets uploaded to the database, the Database class would reject the query and throw an error. This is because I forgot to sanitize my inputs which was very embarrassing. Sanitizing inputs is when you remove or “make safe” ALL the data a user can enter into your database to prevent code injection and MySQL injection. It turns out that when I updated to the project to use the new database class I forgot to re-add the sanitization function. Most of the issues people discovered in my project were quite minor so I’d work on them in the future, but this was such a critical issue with such a simple fix I fixed it immediately. I simply added a new function to the database class which makes any possibly dangerous user data safe.

## Part (B) – Evaluation of success criteria

When I was coming up with my success criteria during the analysis section of my project, I came up with one main objective, and six success criteria. First I’ll list the success criteria, the I’ll evaluate the six success criteria against the finished project and then finally I’ll compare the finished project to the main objective.

### Main objective

“Can I produce a useable and intuitive solution that provides users with an alternative to StackOverflow, while providing additional social and career features not found in mainstream programming sites”.

### Success criteria

- User friendly and intuitive

- Functional and robust question and answer system

- Functional and professional career system

- Functional tutorial zone

- Fast and reliable database and back-end

- Secure back-end

### User friendly and intuitive

This was quite an important success criteria for me as its very important for a user to have a good first impression when they use a website for the first time. In my analysis I said I’d be able to quantify this by getting a class of younger students to test the project. 86% of the testers who responded to my survey said that they liked the design of the website, which I think makes the project meet the user friendly success criteria. 71% of people also said they had no problem trying to use any of the features, which is a big enough percentage for me to consider this criteria met.

### Functional and robust question and answer system

In my analysis I split this success criteria up into subsections as I felt it was to broad and vague on its own. These subcategories and an analysis of each of them are listed below:

|  |  |  |
| --- | --- | --- |
| **Success criteria** | **Met/Partially met/Not met** | **Notes** |
| User can ask questions | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can view asked questions | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can search for questions | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can filter questions by the three categories “Hot”, “Top” and “New” | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can tag questions with the language they are about | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Users can filter questions by their language tag | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Users can comment on questions | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can edit comments | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can vote on questions | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can report inappropriate questions | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Users can save their favourite questions | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Users can answer questions | Partially met | Answer system merged into comment system for a more forum style approach |
| Users can edit answers | N/A | Answer system merged into comment system for a more forum style approach |
| Users can vote on answers | Partially met | Answer system merged into comment system for a more forum style approach |
| Users can accept answers | Partially met | Answer system merged into comment system for a more forum style approach |
| Users can report answers | Partially met | Answer system merged into comment system for a more forum style approach |
| Users can comment on answers | Partially met | Answer system merged into comment system for a more forum style approach |
| Users can report comments | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |

Overall I think that I met enough of these subsections to class this success criteria as passed. One thing I ended up doing differently during development to what I drew up in my analysis was merging comments and answers. This is due to more research into StackOverflow, which has both comments and answers, which I feel makes the whole process more complicated than it needs to be. Merging them also cut down the already extensive development time.

I feel that all the subsections that were fully met were well executed and worked exactly how I intended them to. I was particularly pleased with the design and functionality of the Q&A system as it was the first section of the website I completed and showed me that the time I’d spent planning and designing it has not been wasted.

### Functional and professional careers system

This criteria is also split into categories. An analysis of each of these categorises is below:

|  |  |  |
| --- | --- | --- |
| **Success criteria** | **Met/Partially met/Not met** | **Notes** |
| Employers can create adverts for jobs | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Employers can remove adverts for jobs | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Adverts will be taken down once the position is filled | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Employers can view analytics data for their advert | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Employers can tag jobs | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Employers can specify job location | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can report jobs | Not met | Users can apply for jobs if the employer adds their contact details (email address/phone number) to the job description |
| Users can apply for jobs | Met | Users can apply for jobs if the employer adds their contact details (email address/phone number) to the job description |
| Users can have their job displayed on their profile | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |
| Users can contact employers | Met | Users can apply for jobs if the employer adds their contact details (email address/phone number) to the job description |
| Users can search for jobs based on a search query | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can search for jobs based on tags | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can search for jobs based on location | Met | Proved by both my white box tests and the functionality tests performed by independent testers |
| Users can view jobs on an interactive map | Not met | Not met due to time constraints. This feature will be added in the future for improved user experience |

I was happier with how I met the success criteria for this section than in the Q&A section. I met more of the subsections and I think the career system is a more polished section of the website than the question and answer system. I also enjoyed the programming for the careers more than any other area of the site because it involved some interesting problem solving and approaches to problems I couldn’t imagine when I was planning the project.

The responses from my survey back up my opinions on the careers system. 86% of people said they liked the design, 71% of people said they had no problem using any of the features, and 95% of people thought at least most of the features were easy to use for the first time. These are all high percentages which I think

Adding an interactive map would’ve been a very simple task program using the free Google API but I didn’t add it because of the complexity it would add to both the site and the write-up. This is the main reason for not adding seemingly simple features to the project.

### Functional tutorial zone

This criteria is also split into categories. An analysis of each of these categorises is below:

|  |  |  |
| --- | --- | --- |
| **Success Criteria** | **Met/Partially met/Not met** | **Notes** |
| User friendly UI | Met | Proved by external testers during black box tests |
| Good UX design | Met | Proved by external testers during black box tests |
| Detailed and easy to use tutorials | Met | Proved by external testers during black box tests |
| No dead links | Met | Passed at time of writing |
| No spelling mistakes | Met | Proved by external testers during black box tests |

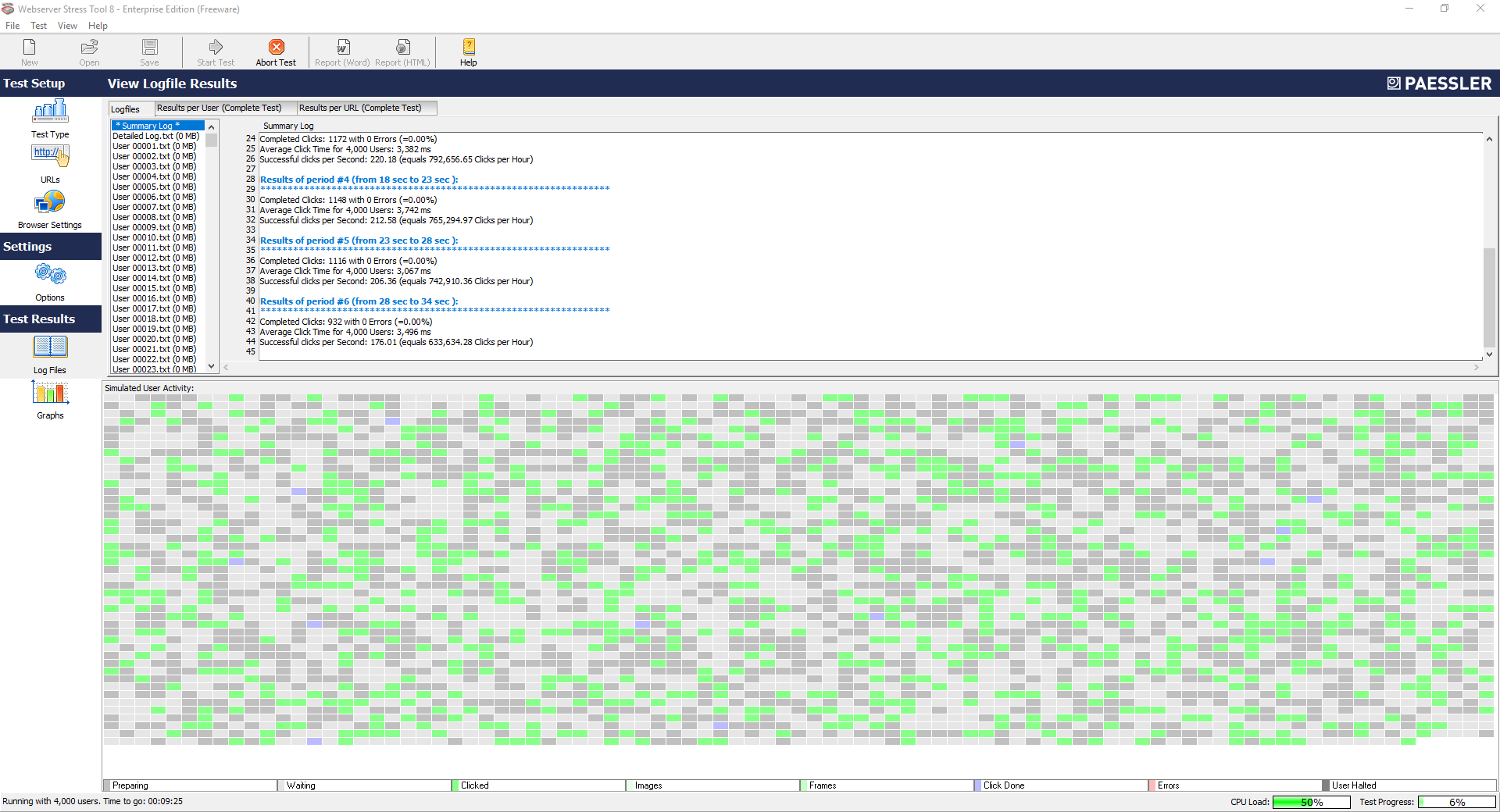
Since the tutorial zone was the easiest section to write and contains only markup, this had the least success criteria and all of them were met. It was also the easiest to test because every subsection could be tested during my black box tests using my survey. 86% of people agreed that the tutorials were easy to follow which is a very high percentage, and another 86% of people said that they liked the design of the website. This is the category that passed the most success criteria and I feel that it’s a very strong addition to the site.

### Fast and reliable database and back-end

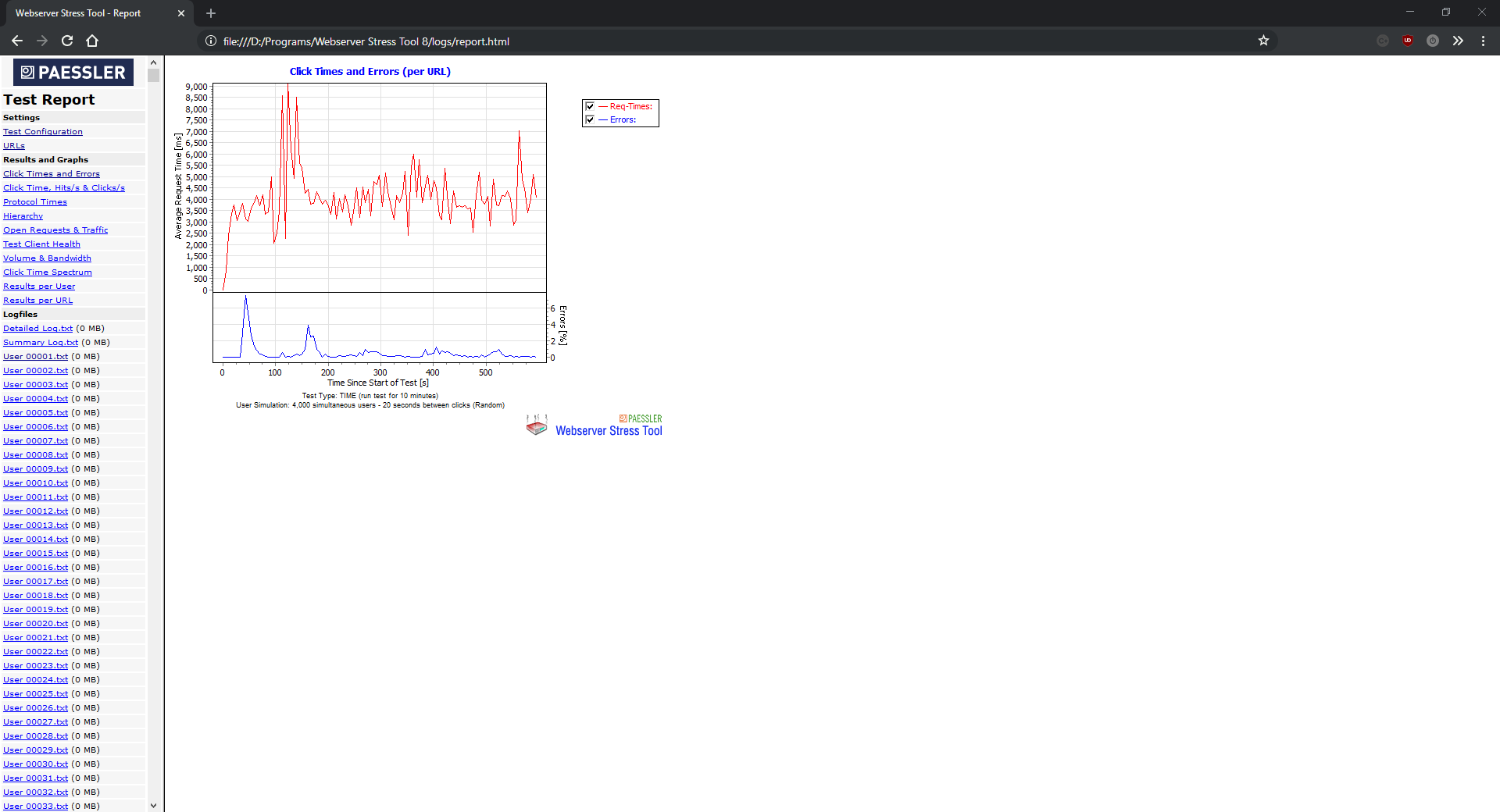
Unlike all the previous success criteria, the next two are not broken up into subsections as they are quantifiable themselves. I wanted to have a fast and reliable back-end system and database to keep the website running smoothly. I also wanted to keep the cost down as I was paying for everything myself, so I chose a relatively inexpensive Digital Ocean Ubuntu Virtual Machine to run both the web server and the database. The peek usage on the site was during the black box tests with over 30 concurrent users, and the website was fast and smooth to use. In a production environment however, there could a much larger number of concurrent users which would be an issue for the cheap server. To solve this, I’d upgrade the hardware the website is running on if the website started to become more popular. This would be easy to do because of the way I’ve set up the deployment of the website. I simply run a script to install the packages needed to setup the server, then clone the git repo into the apache public directory.

To test how fast the current server is, I performed a load test on the website using a tool called Load Impact (loadimpact.com). This test lasted 10 minutes and had up to 135 requests per second. The minimum time taken to perform a request was 3ms and the average was 43ms. There were no errors. 43ms is a perfectly acceptable response time for the server, and shows that the back-end can cope with well over 135 requests per second. For this reason I consider this criteria met. I also used another tool called Webserver Stress Tool 8 as it allowed me to test up to 4000 concurrent users for 10 minutes, a much higher number than the original test. At 4000 concurrent users, the server was really starting to struggle. 0.5% of requests didn’t complete due to a load induced error, and average request time went up to 4000ms. This is an example of a use case where I’d upgrade the server to make it faster and more reliable, however my project isn’t going to get 4000 concurrent users, so this is more of an academic test. The results can be seen below:

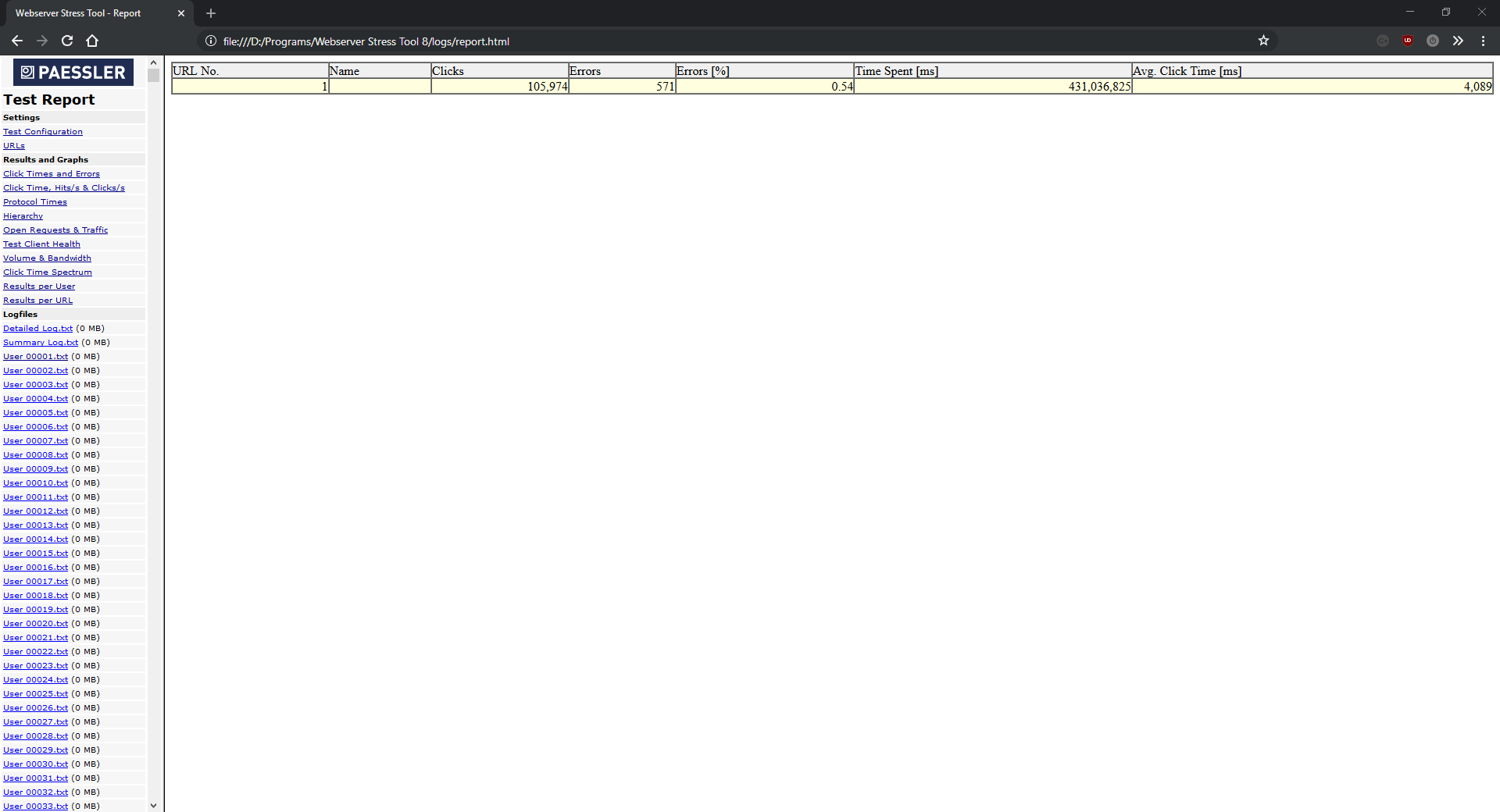
Test running:



Test results:



More test results:



### Secure back-end

This is hard to quantify as security vulnerabilities can come from anywhere. In my analysis I said that if no one had managed to gain access to my project during it’s development I would consider this success criteria met. No one managed to gain access to my project, however during my black box testing, several people accidentally discovered a vulnerability that would have allowed anyone who knew what they were doing to perform SQL injection on any of the login pages. This is because I forgot to re-add the function to sanitize user inputs to my database class. This meant that when the testers were typing in usernames that contained apostrophes, it would end the MySQL query and then anything afterwards could be ran against the database. This is a massive security vulnerability, however as soon as I saw the feedback from the tests, I re-added the sanitization function to the database class to “make safe” all data entered by the user. Without this function this criteria wouldn’t be met, however since I managed to add it so quickly after the test results, I consider this success criteria to be met.