Evaluating Software Using Black-Box and User Testing

Oliver Levay, Kristina Sedelius, Adam Tegelberg, Emma Haggren
Lunds tekniska högskola
Lund University
Lund, Sweden
(ol1662le-s, kr0363se-s, ad3444te-s, em5261ha-s)@student.lu.se

***Abstract*—This research project compares automated testing, using the software *Cypress*, and manual human testing. This was done by creating three versions of a simple login page as an application and creating a quality assurance checklist relating to the quality attributes described in ISO/IEC 9126. The different versions each had a random set of faults related to the quality attribute requirements that were supposed to be fulfilled according to the quality assurance checklist. After creating the website, tests were written in Cypress using the checklist. The developers that created the website and that wrote the tests had very limited communication to ensure that the tests were not influenced by the implementation. Four tests subjects used the checklist to identify issues with the login page. Automatic testing was more reliable than human testing, a computer will not get tired or have opinions about what is worth testing. The human test subjects produced feedback that was never even considered by the research team. Automated tests written in Cypress might not have this ability. The results of the testing showed that the automated tests were more accurate and consistent. Furthermore, the results suggest that a combination of automated tests and human user testing in order to get the best of both worlds with the current tooling available. Evaluating if the detailed feedback that the human test subjects gave is a unique human trait was outside the scope of this paper. The team was surprised by how well automated testing could find faults in usability, and frustrated how hard it was to write automated usability tests.**

I. Introduction

A. Background

Software quality assurance is crucial to ensure that applications meet user expectations and standards. This project focuses on evaluating a simple login page against the ISO/IEC 9126 product quality standards, which include the attributes functionality, reliability, usability, efficiency, maintainability and portability [**?**].

B. Objective

The goal of this research project is to compare automated Black-Box testing with human testing. The automated tests were written in Cypress and both the automated tests and the human tests were guided by a quality assurance checklist. We aim to answer the following research questions:

1) What is the overlap between the software quality issues that can be identified by human testers and those that can be identified by automated Black-Box testing?
2) Which types of software quality problems can only be found by human testers, and which can only be identified by automated Black-Box testing?
3) Which types of issues can't be identified by either method?

II. Method

A. A login page

A login page was chosen because of these main reasons.

1) It's very common on the web, which means that all users have preexisting conceptions of how it should work.
2) It's not too complicated to make.
3) It's very important that it works as the user would expect, and that it works well, or the user will not be able to even try the product or service. The team believes that if the login page does not work all other type of testing is irrelevant. This makes the login page the most important part of a web-based business

B. Software Implementation

Multiple versions of a simple login page was created using React. The versions were created with intentional issues connected to the ISO/IEC 9126 standards. At first a simple and functioning login page was created, then five new versions were created. Each initial version had a unique focus on exclusively functionality, reliability, usability, efficiency or portability. The introduced issues were:

Functionality:
- Faulty logic for login-check, for example not requiring a password, or only being able to log in with admin and no other users.
- Password is not case-sensitive.
- Enter key disabled for logging in.

Reliability:
- A pop-up when the login failed, that was irremovable preventing recoverability.
- A ten percent probability of randomly failing.

Usability:
- Non-centered components.
- Text with small size, Comic Sans font, and low contrast from background color.
- Password- and username-field were switched.

- Cursor disappears when hovering over the login button.
- Overall visually displeasing.

Efficiency:

- A delay caused by extremely inefficient search algorithm.
- Multiple confirm pop-ups before attempting a login.

Portability:

- Unable to login on Firefox, Safari and Edge. Only usable with chrome.

Maintainability was excluded because it cannot be evaluated with either Cypress or user testing, as it requires analysis of the internal structure of the code. The team that are supposed to write the test cases, and the human test subjects are not supposed to have access to the code rendering the evaluation of maintainability impossible. After the versions were completed they were merged into two versions with mixed issues. The reason for this was so that a tester would not be able to guess which other issues the versions had based on the previously identified issues. For example if a test person notices a pattern of functionality faults and the previous version only had usability faults, they can guess faults without actually testing. The tests were formed so that the faults were random between the different quality criterion.

At the end of the development the final versions were the one original login page, and the two faulty versions. Three users were created with credentials that was communicated to the testing team and the test subjects. The credentials are:

Username	Password
admin	12345
user1	Password1
user2	Password1

Fig. 1: Credentials

In addition to the login credentials, the testing team also received a list of elements and their corresponding HTML ID. This was to ensure that the tests would be able to identify the correct element. The elements and their corresponding HTML ID are outlined below.

Element	HTML ID
Login Header	login-header
Username Label	username-label
Username Field	username
Password Label	password-label
Password Field	password
Login Button	login-button

Fig. 2: Elements with ID

The team that produced the website code and produced the test code had no further communication related to the code or the tests. The team that created the website had no knowledge of the tests and the team that wrote the tests had no more knowledge of the website code than what is outlined above.

The versions that were tested are outlined below:

- **Version 1**: The original version where all the criteria were met. As seen in figure **??** and figure **??**.
- **Version 2**: A seemingly proper version with hidden issues that included a number of problems, but mainly some functionality and portability problems. As seen in figure **??** and figure **??**.
- **Version 3**: A clearly visually displeasing version focus of this version was to make the site break usability, unreliable, inefficient and without some basic functionality. As seen in figure **??** and figure **??**.

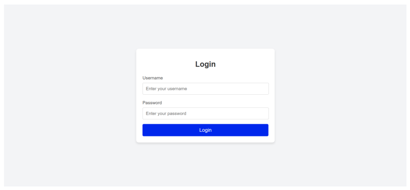

Fig. 3: Version 1 and 2, Full sized window

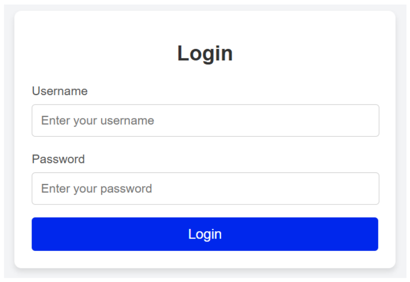

Fig. 4: Version 1 and 2 zoomed into login component

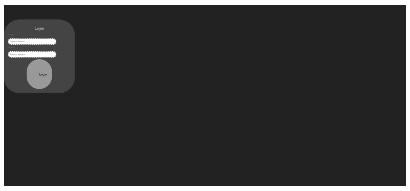

Fig. 5: Version 3 Full sized window

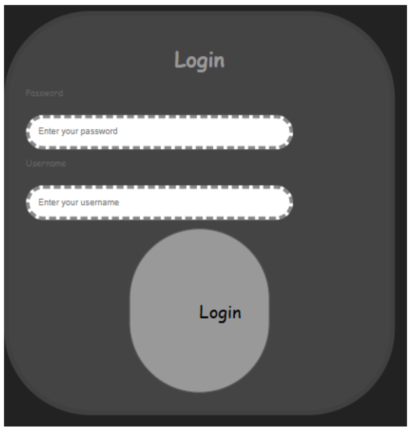

Fig. 6: Version 3 zoomed into the login component

The produced website code is available in its entirety in appendix C.

C. Checklist Development

The quality assurance checklist was developed and categorized using the ISO/IEC 9126 software quality standards and their subcategories.

The checklist included the following criteria:

Functionality:

1.1 Verify that a user can log in with a valid username and valid password.
1.2 Verify if a user cannot log in with a valid username and an invalid password.
1.3 Verify that it says "Invalid username or password!" somewhere for invalid login.
1.4 Verify if the data in password field is either visible as asterisk or bullet signs.
1.5 Verify if the 'Enter' key of the keyboard is working correctly on the login page.
1.6 Verify that the login button attempts a login.
1.7 Verify that logging in will the take the user to /LoggedIn
1.8 Verify that a user that is logged in to the correct account.
1.9 Verify that the password is Case-sensitive

Reliability:

2.1 Verify that it is possible to log in again after an error.
2.2 Verify that logging in behaves the same way every time.

Usability:

3.1 Verify that the font is easy to read.
3.2 Verify that the contrast of text color and background has a high enough contrast
3.3 Verify that the layout of the login page is as you would expect.
3.4 Verify that the labels are useful.
3.5 Verify that the page is not really ugly.

Efficiency:

4.1 Verify the time taken to log in is less than 1 second.
4.2 Verify that logging in is not unnecessarily complicated.

Portability:

5.1 Verify that the login page works on Google Chrome.
5.2 Verify that the login page works on Microsoft Edge.
5.3 Verify that the login page works on Safari.
5.4 Verify that the login page works on Firefox.
5.5 Verify that the login page works on a smartphone.

D. Automated Testing Using Cypress

Cypress was used to automate tests that evaluate the quality of the login page against the checklist criteria. The tests for *functionality* mostly consisted of simulating writing the username and password and testing if the website behaves as expected.

The tests for *reliability* related to the capability of software to maintain its level of performance over time. This includes recoverability, the ability to recover from errors and reliability compliance where the login page is expected to work the same every time. To test recoverability, tests were made to intentionally produce an error by typing the wrong password and then attempting to log in. Reliability compliance was tested by logging in 50 times.

Usability testing was automated by making assumptions about what it means to have an expected layout and a nice looking website. The assumptions were that an expected layout means that elements should come in the order of username, password and then button. A login website that is "not really ugly" was assumed to have readable text, an expected layout, centered elements and appropriate spacing between elements. The tests that made sure that the elements are centered on the screen also made sure that they were aligned with each other. Label usefulness was tested by making sure that the text above username says something close to "user" and the same for the label describing the password input field.

Efficiency testing was automated by measuring the time between starting a login attempt and the login screen appearing. In addition to the time it takes for the software to complete its task, tests were also written in a way that ensured that the login process was not too complicated for the user. This was done by ensuring that there were exactly two input fields. If there are more input fields one can assume that the login page is too complicated and time-consuming for the user. Furthermore, the tests ensured that there was exactly one field responsible for the username and exactly one responsible for the password.

Portability was tested by running all the tests again for all the different browsers in the specification.

E. Hypothesis

Before testing began one hypothesis was that some issues, such as subjective usability concerns, would not be detectable by Cypress. Another hypothesis was that humans are error-prone in their nature and that this would hurt their consistency in finding errors across several people. The team were not sure which categories of issues would be hard for a human to test. If a user has a Windows computer testing Safari is exceedingly difficult, but it is not impossible. The research team had an idea that the test subjects would believe it impossible and not attempt to find a solution to this problem. The team had high confidence that the assumption that some issues can only be identified by humans and some can only be identified by a computer was true. We believed that the testing would result in an interesting Venn diagram outlining the overlap, intersection and difference of the sets of faults that can be identified by either method.

F. Human Testing with Real Users

Participant Selection: Four people participated in the testing in total. All test subjects study computer science or information and communication technique at LTH. One of the test subjects participates in the course that this research project is connected to; a course called 'Software Testing' at LTH.

Testing Procedure: The finished website code was uploaded to a cloud service called Netlify so that the test subjects could display it on their own devices. The test subjects were given a paper with the scenario, a list of the valid login-information and the quality assurance checklist. The exact scenario given is available in appendix A. The participants were then asked to follow the checklist's criteria and evaluate if it was true for the given version of the website. The participants followed through with the checklist for each of the three versions, all the while ticking off what true, and circling what was false. If the test subject was unsure they left it blank.

III. Results

A. Test Creation

Using Cypress to create automated testing turned out to be mostly straightforward and not too complicated. There were a few hurdles that we had to overcome.

1) Cypress does not support Safari. There is experimental support for Safari using the open source browser engine WebKit which Safari is built upon. This means that the tests are not run in the actual Safari browser. This could potentially lead to differences in how the website is displayed and how the tests are run.
2) The usability tests had a lot of complicated code to compare layout alignment, ordering and color contrast. Cypress does not natively support this kind of testing. Testing how long something takes or simulating pressing a key on the keyboard for example has native support which makes that kind of testing easy and efficient. Usability testing turned out to be possible but not easy.

The test code is available in its entirety in the appendix B.

B. Automated Testing Results

The results from running the test have been compiled to 4 reports, one for each browser supported. The reports are available at appendix E. After the test reports were compiled the team used them to determine the checklist validity. The research teams interpretation of the results are outlined in its entirety in appendix E. In order to compare automated testing with human testing the results from both are compiled in a Google Sheets document available in appendix D. The accuracy of the answers are outlined below.

1) Fully functional: All tests passed which lead the team to believe that all the requirements are fulfilled. This was also the correct answer according to the keys that the team had prepared before initiating testing.

2) Mostly reliability and portability issues: Checklist item 3.1 *Verify that it is possible to login again after an error* was incorrectly marked as TRUE. In the testing code an error is produced by attempting to enter an incorrect password, but in this version every password is correct, so no error is produced. This issue would be remedied by creating a test that produces an error by entering an incorrect username as well.

3) A lot of issues: Surprisingly version 3 that we thought would be difficult for the automated testing suite to evaluate did not have any mistakes. The tests would suggest that the labels are useful because it says ”Username” and ”Password”[1] on them. The tests show that the font is not a usual one and the font size is too small, and the contrast is bad and hard to read. The team made the call that the labels are not useful based on this information.

In summary all the issues that we set out to test *can* be identified with automated testing.

C. Human Testing Results

: The results from the human testing was inconsistent, but all issues had at least one correct answer from one of the test subjects. The humans were terrible at identifying Portability issues because if they did not have the browser requested they did not make efforts to test it on a friends computer or install a new browser. Some test subjects assumed that it would work on the other browsers because they could not be bothered to test it out, or did not answer. Other test subjects mistakenly assumed that the site worked on a given browser because it loaded correctly. If they had attempted to log in they would find out that it does not work. The automated tests ran by the computer had no issue running all the tests again with a different browser. Boredom and laziness is not a factor for the computer. For version 3 one test subject could not log in because they did not try the admin account which gave wildly inaccurate results and tracking the biggest fault would be difficult. The computer did not have any issue logging in with each account.

D. Analysis of Overlap and Differences

Let A be a set of issues found by automated testing and B be a set of issues found by human testing. Using some set operations we can answer our research questions.

1) Union $(A \cup B)$ Issues identified by either method.: There are no type of issues that cannot be identified by either automated tests or human testers because automated tests *can*can identify all types of issues we set out to find.

2) Intersection $(A \cap B)$ Issues identified by both methods.: Since at least one human correctly identified each issue. The overlap between the two methods is 100%.

3) Unique to either (A \B) Issues only detected by automated testing.: This is an empty set since all issues *can* be identified by either methods.

4) Unique to Human Testing (B \A) Issues only identified by human testers.: This is an empty set since all issues *can* be identified by either methods.

5) Union complement $(A \cup B)^c$ Issues not found.: This is an empty set since all issues *can* be identified by either method.

E. WebKit vs Safari

The testing team was worried that WebKit and Safari would not behave the same way during testing and would for example have different User Agents. According to the Mozilla Developer Network the HTTP User-Agent request header is a characteristic string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent.

To determine the user agent the testing team used the following code:

```
let isSafari = /^((?!chrome|android).)*safari/i.test
    (navigator.userAgent);
```

Listing 1: Detect Safari Browser

When asserting if this is true or not it was true in WebKit but false in Chrome, Firefox and Microsoft Edge. This signifies that the user agents are the same on WebKit and Safari.

Playwright is the engine under the hood of the Cypress testing suite. According to the official Playwright documentation, running WebKit simulations on a Mac is very close to the real experience, but some issues arise if you run it on the most affordable option of a Linux system. For example video playback may not behave as expected. This happened to not be an issue for the testing team since one of the team members have a MacBook and one of the team members have a Windows computer. Running WebKit with Playwright on a MacBook is stated to be as close to running on closest-to-Safari experience as possible according to the documentation.

IV. Discussion

A. Creative Feedback

The results of the testing prove our hypothesis about the strengths and weaknesses of both automated and human testing to a strong degree. However, some new key insights were also found. The automated tests were more accurate in finding issues on the login page with no human finding all problems. The automated tests could not, in contrast to human testing, give any creative feedback. The user feedback obtained during the testing gave us new insights about our program that the team had not thought about in cases even altered the opinion of the team of what the correct answer to a problem was to begin with.

The human testing had a large variance in the problems found whereas the automated testing always gives the same results. The variance could be explained by many factors, since the test subjects are different from each other in very meaningful ways.

B. Human Motivation

The test subjects in general lacked motivation. Results from the portability testing show that people generally do not install a new web browser for test purposes. Nor did the test subjects try to use their friends' computer. Some test subjects did not care to test it even if they had the correct browser installed. Even the ambitious and motivated test subjects felt done after testing if the login worked at all, whereas the computer ran all the tests again on all the browsers without complaining. Most test subjects had low motivation, probably due to a lot of factors but most importantly the following:

1) They do not get paid to do the work, but they do the testing out of their own good will.
2) Most test subjects did not find satisfaction in repetitive tasks. A good test subject who enjoys the task might do better.

We did not have the resources to spend more time in order to find test subjects that enjoy repetitive tasks. Additionally, we did not have the resources to pay the test subjects with anything else than gratitude. If we were to retry this experiment with more resources we attempt to fix both of those issues. The problem with putting too many resources to compensate for human weakness is that a computer does not have these problems. One could argue that makes the factor irrelevant. The computer will never question the importance of testing and will always try to do as they are told.

C. Results vs Hypothesis

The team honestly thought that there probably are some issues can only be identified by humans and vice versa. Furthermore, the team believed that there were some issues that would not be identified by either method. This turns out to also be false. The Venn diagram that would be produced by the results that we found would not be interested in the slightest. All the types of issues can be identified by both methods, the diagram would be a mono colored block. However, the computer definitely has the edge on most kinds of testing, but humans can exceed the expectations of the person writing the tests. This may be ratified as well using AI to ask for subjective opinions and suggestions. When asking popular AI chatbot ChatGPT whether the first/second version or third version looks better it responds like this: ”The first image (version 1/2) is aesthetically more appealing according to modern design standards. It follows the principles of minimalist design with clear lines, good contrast, and readability. It uses a light background and a blue button to highlight the most important interaction (the login button), making it more user-friendly and professional.

The first image feels more playful but less modern and professional. The dark background and outlines can give an overwhelming impression, especially for users who prefer simplicity and clarity.

If the goal is to create a more user-friendly and modern experience, the second design is preferable.”. This suggests that modern AI can be used to make reports on subjective issues.

D. Exceeding Expectations

Writing tests in Cypress will only be as good as the faults that the people writing the tests can imagine. The only real issue that the computer did not find was that in version 2 was the missing recoverability. When writing the tests the testing team assumed that writing the wrong password would produce an error. This shows that the accuracy and reliability of the test is limited by the knowledge, imagination and skill of the test writers. Using modern tools like ChatGPT the test writers can mitigate this factor somewhat, but we believe that no matter how good you are some things will inevitably be missed. One could however argue that this fault was irrelevant because if writing the wrong password does not produce an error recoverability is a minor issue until that is fixed. If the security issue were to be fixed then the tests would work as intended. Additionally, some humans made the same mistake because in version 2 there is a risk of random failure every time a login is attempted. This error is possible to recover from which confused the humans and made some of them think that every fault would behave the same. Some test subjects were initially confused when attempting to produce a fault since every password worked. After a short thinking period all test subjects came to the conclusion that they should try to change the username, Cypress can not deliberate and alter their testing methods based on unexpected results.

E. Writing automated Usability tests

Writing usability tests is much harder than the other kind of tests. The reason it was hard was because we had to write a lot of the logic ourselves. We believe the tools for this kind of testing is not included in most testing suites due to the commonly held belief that running usability tests with a computer is fruitless because only humans can do accurate testing. This turned out not to be true.

As it stands right now writing Usability tests seem to not be worth it due to the difficulty, but we believe that this is an issue with insufficient tooling and not a problem that is unsolvable.

F. Recommendations

With the existing tooling that is available today, using a combination of automated tests together with human user testing yields the most accurate test results. The automated tests have greater accuracy and are easier to perform, and testing with humans gives more depth and could help make the tests more effective with creative feedback.

G. Limitations of the project

The login website is a quite small website with only one functionality, to log in. This creates a testing scenario with a very small scope, and the functionality is isolated. In more common websites, a login page would be a small part of the website and would need to be integrated with the rest of the application. This means the testing was a bit simpler, and might not be able to show how human testing compares to computer testing on a larger scale when it comes to efficiency. The testing can however show patterns in what type of criteria human testers or computers, lack in or excel at testing. We picked a login page because of these main reasons.

1) It's very common on the web, which means that all users have preexisting conceptions of how it should work.
2) It's not too complicated to make.
3) It's very important that it works as the user would expect, and that it works well, or the user will not be able to even try the product or service. We believe that if the login page does not work all other type of testing is irrelevant making it the most important part of a web-based business

V. Conclusion

A. Summary

In conclusion, automated tests proved more robust and highly reliable, additionally Cypress was able to find the intended design errors; something that was unexpected. The human testers provided inaccurate and inconsistent results as well. The automated testing did not give feedback directly on the tests or application itself, and human testers gave significant details that previously were not realized. We have found that human testing might provide something that an automated test cannot but in that case it is something that we did not look for. Finding this out would require further research.

VI. Future Work

A. Is human creativity unique?

We have found something that humans were better than our tests at that we did not expect to find. Is this unique to humans or is it possible to do the same with AI? Early research like the ChatGPT example from earlier suggest that it is not unique to humans. We recommend further research to answer the question ”To what degree can human creativity in giving feedback in user testing be reproduced by automated Black-Box testing?”. Our hypothesis is that most if not all the human strengths are not unique to humans. Creating a testing suite that gives this kind of feedback would be a valuable tool. One could very easily use Cypress to automate this process. Here is a suggestion to how this could be done:

1) Take a screenshot with Cypress
2) Send the screenshot to an AI, and ask if it is visually appealing TRUE or FALSE
3) Assert that the answer is TRUE.

This might make it possible to test subjective opinions with Cypress. By sending screenshots with each interaction you may get good spontaneous and unexpected feedback from the AI as well. It would be interesting to see if the AI generally gives better feedback than humans, and if it would be cheaper to run automated tests even when compensating for the cost of the test developer and the cost of using an AI. The AI requirements are quite resource intensive, the most costly is the image recognition and sending images to the AI. Our hypothesis is that AI would give better and more consistent feedback than humans and that this method could potentially entirely replace human testing the unit, integration and system level. Acceptance testing will always have to be done by a human because testing in theory and on a real phone can be very different. Another hypothesis is that the feelings that users get when using a product can not be accurately simulated and estimated by an AI.

B. Can a website be ugly and useful?

Our findings suggest that there is a tight link between beauty and usefulness. Is there such a thing as an ugly website that is easy to use? Would it be generally easier to use if it was more pretty? Would it be more pretty if it was easier to use? Our hypothesis is that they are strongly linked, but an interface that is ugly can still be easy to use, but making it easier to use would also make it prettier at the same time.

C. Is Playwright + WebKit an accurate representation of Safari?

We found that Playwright and WebKit was very similar to Safari, but there are things that probably behave differently. Would it be possible to create a test case in Cypress that fails on Playwright + WebKit but would succeed with Safari? The main difficulty with this further research is that you cannot use Cypress with Safari which is the entire reason why we used WebKit to begin with. One would have to either use some other suite that might have Safari support, or do the testing manually.

D. *Including maintainability*

Maintainability was excluded from the research because we wanted to specifically focus on Black-Box testing. Evaluating whether humans or computers are better at evaluating maintainability would require White-Box testing methods.

E. *Evaluate effectiveness*

This project only tried to find if it was possible at all to identify categories of issues. The results suggest that it would be more research is required to determine the degree to which humans are better at one kind of testing compared to another since it turns out that all categories can be identified by either method.

VII. Contribution statement

Oliver: Project planning, report writing, produce the checklist, compiling the results from the user testing, produce automated tests, run the user tests, interpret the test results, publish the website to the cloud, prepare the scenario, run the tests and compile reports from the results.

Kristina: Project planning, report writing, produce the checklist, compiling the results from the user testing, produce the website code, run the user tests.

Adam: Project planning, report writing, produce the checklist, produce automated tests.

Emma: Project planning, report writing, making the checklist, compiling the results from the user testing, produce the website code, interpret the test results.

References

[1] ISO/IEC 9126, Software engineering — Product quality, The International Organization for Standardization, 2001-06.

To save space all of the appendices are links. All links have corresponding QR codes.

Appendix A
Scenario

The user scenario that was presented to the human test subjects: https://bit.ly/scenario-user-testing

qr-codes/scenario.png

Fig. 7: Appendix A: Scenario

Appendix B
Cypress Test Code

The test code that was run to produce the automated test results. https://bit.ly/cypress-code

qr-codes/cypress-test-code.png

Fig. 8: Appendix B

Appendix C
Github Repository

The GitHub Repository in its entirety which includes the code for the different versions of the login page. https://bit.ly/complete-repo

qr-codes/github-repo.png

Fig. 9: Appendix C

Appendix D
Results comparison

A Google Sheets document comparing our key, the computers answer and the degree to which all human test subjects answered correctly for each version. https://bit.ly/results-comparison

qr-codes/results-comparison.png

Fig. 10: QR code to the appendix D

Appendix E
Automated test reports

The test reports that were generated after running the automated tests. https://bit.ly/results-automatic

qr-codes/results-automatic.png

Fig. 11: Appendix E