Adding Visually Impaired Accessbility Features into HEAT

Elaine Loke
School of Computing
University of Kent
Canterbury, Kent
el345@kent.ac.uk

Hannah Cooper
School of Computing
University of Kent
Canterbury, Kent
hc421@kent.ac.uk

James Wong
School of Computing
University of Kent
Canterbury, Kent
cw692@kent.ac.uk

Joanna Oyegbile
School of Computing
University of Kent
Canterbury, Kent
joo29@kent.ac.uk

***Abstract*— This report outlines the work that was done to bring accessibility features to HEAT, specifically for visually challenged people who are colorblind. Colour blindness is a vision impairment caused by a genetic mutation that changes the subject's colour vision by lowering sensitivity to certain colour wavelengths. The method we used to approach these implementations is the Agile and Scrum method. Researchers have recommended that the Agile and Scrum methods are the most dependable when it comes to project management, as the objective is to improve the speed of development. Using this method we were able to successfully implement into HEAT the accessibility features that we aimed to do such as the contrast theme selectors for the users and the ability to adjust the font size of the entire HEAT software.**

Keywordst; Visually Impaired; Colour blind; Contrast; Font Size

I. INTRODUCTION

In this report we will be talking about how we implemented features into the given software HEAT, to ensure that it is more accessible for visually impaired individuals who are colour blind. According to the [15] [21], virtual impairment can be either the complete loss of vision or partial loss. It does not only include partial, or complete blindness; but it also includes visual impairments such as; extreme light sensitivity and colour blindness [16]. Colour blindness is described as a vision deficiency caused by a genetic mutation that alters the colour vision of the subjects by decreasing the sensitivity to certain colour wavelengths [18]. Allan, Kirkpatrick and Henry (2016)[1] explains that there are a variety of forms when it comes to colour blindness such as: Red - Green Colour- Blindness (Deuteranomaly, Protanomaly, Deuteranopia and Protanopia), Blue - yellow Colour-Blindness (Tritanopia and Tritanomaly) and Monochromacy (black and white).

Research suggests that people who have any of these colour blindness will have difficulty identifying colours of similar colours or certain shades [13]. Therefore, this project report aims to focus on the accessibility needs for individuals with colour blindness as a visual impairment. In this report we will highlight: what accessibility feature we chose, why we chose colour blindness accessibility as a feature we would like to implement; we will also describe the method and the developmental process that were used to implement these features; as well as discussing what went well and the limitations we encountered when modifying the HEAT software to implement colour blindness accessibility.

II. ACCEPTANCE CRITERIA

We opted to offer more splash windows with more configurable options after talking with the customer. Our theme selector window had to be dismissed using the button in the top right corner at first, but we added a button to allow the selected theme to be applied and to close the selection window easily. This matched the customer's request and backs up our first user story. The initial window options for theme selection are all suited for someone with red-green colour blindness, and there are further possibilities after opening the application. The design extends into the option menus, making HEAT even more accessible. The ability to adjust to a more comfortable font size was also added.

III. METHOD

A. Justification of Methodology

Initially we looked at what methodological process would be best to ensure that we get a high level of functioning implementations in HEAT; and after further research on the type of methods used in software project management, we decided to use the Agile and Scrum method [20][4]. The Agile and Scrum method has been suggested by researchers to be the most reliable method when it comes to project management, as the aim of this is to increase the speed of development, and an aptitude to respond to change through effective communication [19]. Therefore, because of the time limitation of this project, we decided to conduct a one week sprint focusing on the features that are most important in our implementation to coincide with our user stories [5].

Furthermore, we conducted a group meeting before the start of the one week sprint to make a decision on which type of visual impairment we will be catering to and carried this daily meetings on throughout the week [5]. After further research we decided to create a feature that worked with

accessibility for colour blind users. Using the user stories we created for this type of visual impairment, we came up with a plan of action including: what features of accessibility would be useful for colour blind individuals, what approach we are going to take when implementing these features, and research how we will test the accessibility complaints with the program HEAT [9]. In addition, we chose not to implement a text to speak for visual impairment for partial blindness/ blindness because it would not have fit into our one week sprint window.

B. *Evaluation of Methodology*

The advantages of this method is the design ability to respond to changing requirements of the project [11]. This is important because it means that the changes will be integrated immediately and there will be no confusion of communication between the team and its client [14]. However, this method does have some disadvantages such as, if this project was on a larger scale, then it would have been more difficult to judge the effects and the time required for the project in the software development [14].

Furthermore, there are other methodological approaches such as the Waterfall method that is used in software project management [8]. The Waterfall approach involves the sequential process of each stage of development, this means that it begins and ends one stage before the next one can start [2]. This is a limitation because, it makes it a *"set-in-stone"* method, and it doesn't provide that same agility and adaptability like the Agile and Scrum focuses on. Moreover, it makes it difficult to make last minute changes in requirements or design [14].

C. *Version Control*

Version control is the practice to track and manage any changes made to the software code, which is especially useful when working in a team [3]. It is important because it helps teams resolve any kind of problems, and to track every change that is made by each member of the team [3]. Therefore, to help manage each of our processes and progress on the implementation features chosen, we used GitLab for version control. GitLab is a platform that helps teams of multiple developers to work on the same project and to share any feedback or changes without messy integrations of each other's codes [6]. We used it as a way for us to create backups or make any changes that were important to our design, by creating branches of each aspect of the code we each worked on.

IV. Developmental Process

Research explains that communication, especially between the client and the developer is important because it aims to reduce errors; hence speaking with the customer about the precise features they desired was one of the most pivotal moments in our process [7] [17][12]. We realised we were taking a different path because our features appeared to be more catered to the general public of visual impairment, which proved to be less effective than we had intended. Following that, we focused our efforts on improving the product's accessibility by addressing and fixing some of our more critical issues, such as some of its functionality. As a result, we went back through our user stories and made our product significantly more accessible to the user's needs [9]. After our initial interaction, we followed up on our questions about what the customer was looking for to make sure we were meeting their needs.

Furthermore, to make the fonts larger and easier to read, we modified the initial font sizes and window colours. The theme is set to dark to accommodate those with light sensitivity. After opening HEAT, you can also adjust the font sizes from the settings menu, which includes more options than before, such as the text in the menu bar. We included a pop-up with the opportunity to choose between light, dark, or high-contrast themes, as these are the most common requirements from our users [1]. The dark theme is primarily black and white, making it ideal for people with various types of colour blindness, while the program's default setting of high contrast makes it acceptable for those with a decreased sensitivity to light [1].

Instead of using software for issue tracking, we all worked on our branches separately before merging, so our bugs were also tracked separately. During our morning meetings, everything that had a significant impact on functionality was discussed [5]. Before we developed our final branch, we tried to resolve any major defects that would damage the product after the merge, however time limited the bugs we could fix. The ones we were able to correct ahead of time made merging much easier because they were easier to address when the project was in smaller, individual branches, and we didn't break any other code trying to fix them.

V. Testing

A. *Unit Testing*

We performed JUnit testing on a variety of classes, including both original source code and newly introduced classes that we'd written to deliver our functionalities. ThemeManager, AboutWindow, HelpWindow, OptionsWindow, and MainMenu are all included. To identify the occurrence of successes, we used the assertEquals method and System.out.println. All of our unit tests yielded positive results, indicating that our classes and methods are capable of implementing our features.

B. *Functional Testing*

We addressed the customer for functional testing to confirm our user narrative and used the agile methodology to implement feedback on our features and user experience.

Prior to submitting our final submission, we worked on making final adjustments and presenting the prototype to the customer.

C. *Code Inspection*

Individuals who did not work on the specific section of the feature implementation evaluated the codes to discover any potential errors, as our codes were produced by different people.

VI. Discussion

We had excellent communication throughout the week. We all shared our thoughts and checked in on each other, which was especially helpful because some of us struggled with areas that the rest of the group was comfortable doing, and vice versa. The features were also implemented more quickly than anticipated. This allowed us to devote more time to bugs, even if we weren't able to fully resolve them. We also assigned roles efficiently because we completed everything on schedule, as planned, at the beginning of the week.

A. *Limitation*

The time constraint was one of the most crucial issues we faced. We had a number of problems that occurred which took up a large amount of our time to try to resolve. An example of this is a audio JAR file that was required in reading out the keyboard shortcut to the user suddenly became unusable near the end of the deadline. After several tries at trying to look for other ways to get around the JAR file issue, we made a group decision to remove it. This is because, despite spending a number of days analysing and trying to identify the issues, we had a lot of trouble with implementation inside the HEAT window because we were using a few different open sources. Furthermore, we also incorporated a magnifier at first, but it had to be deleted later because it didn't work as planned and caused the other functions to malfunction. We explored the matter for a while, but due to time constraints and the amount of work remaining on the other features, it was determined that the magnifier would be deleted entirely.

If we were to do it all over again, one of the first things we'd focus on would be user stories, as they provide a lot of direction for the project's trajectory. This would have saved us a lot of time at the start because the items we needed to finish first would have been more obvious. Moreover, rather than implementing all of the features at once, we would begin with the easiest jobs and accomplish them before moving on to the more complex coding. We integrated most of the functionality before adding a splash window, but it would have been much easier to include it earlier because it would have taken less time.

Conclusion

We looked at the design, approach, and outcome of incorporating accessibility features for colour-blind people in this paper [1]. Because people with colour blindness have trouble distinguishing between colours that are similar, we decided to use features like the contrast theme to ensure that we have themes that were appropriate for those with that difficulty [13]. We also took special care to accommodate individuals who may be colour blind or have low shade contrast sensitivity. Finally, we were able to successfully implement the font size options and mostly implemented the theme selector, minus the console window, however we were extremely restrained due to time pressure. For future consideration, we would have addressed the issues and looked at reimplementing some of the elements that we didn't have time to enact to a level that guarantees the code's functionality.

References

[1] Allan, J., Kirkpatrick, A., and Henry, S.L., (2016) *Accessibility Requirements for People with Low Vision*. Available at: https://www.w3.org/TR/low-vision-needs/ [Accessed: 03/03/202].

[2] Alshamrani, A., and Bahattab A., (2015) *A comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*. International Journal of Computer Science Issues (IJCSI), Volume 12, Issue 1, No 1. Available at: https://d1wqtxts1xzle7.cloudfront.net/36637147/SDLC-with-cover-page-v2.pdf?Expires=1646229760&Signature=HmlkWpCBOT1YE-4ueKpWyUGSCCTYTKNyYvnGE1hCDY242POl2xs0dPFMA-iSxF6lSBHzkQhILqRZRhyD0wToflpiXx3it66IYwBeaveGg2gpYYXipKhGPbdjj83yTB~NSgm-NIW~R5OxDbhtSk4w1m3PWIgdT0WXMcb~L2r5zeo6Z4yWBuzNV0adON7hLaEhPEHd6m-Q8sc5ntxSg1WfYANiAXW-kN5nZ~lMYMa-Rsx7avP5PwDeaVI0eUZ-Vzc0ZO6zQs86mLmKdeJB2cjo1J6xBl~GiUbG98ybu7umYYK5J9WAAGuNPp5D8pTqOr4rIO2VFd0vjzWu92RwEGvppw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA [Accessed: 02/03/2022].

[3] Blischak, J.D., Davenport, E.R. and Wilson, G., (2016) *A Quick Introduction to Version Control with Git and GitHub*. PLoS Computational Biology, Volume 12, Issue 1. Available at: https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1004668&type=printable [Accessed: 03/03/2022].

[4] efone, H.F., (2010) Understanding Agile Project Management Methods using Scrum. OCLC Systems & Services: International Digital Library Perspectives, Volume 27 Issue 1. Available at: https://www.emerald.com/insight/content/doi/10.1108/10650751111106528/full/pdf?title=understanding-agile-project-management-methods-using-scrum [Accessed: 03/03/2022].

[5] Edeki, C. (2015). *Agile software development methodology*. European Journal of Mathematics and Computer Science, Volume 2 Issue 1. Available at: https://www.idpublications.org/wp-content/uploads/2015/05/Agile-Software-Development-Methodology.pdf [Accessed: 03/03/2022].

[6] Haaranen, L., and Lehthinen, T., (2015) *Teaching Git on the side: Version Control System as Course Platform.* Innovation and Technology in Computer Science (ITICSE), pp. 87-92. Available

at: https://dl.acm.org/doi/pdf/10.1145/2729094.2742608 [Accessed: 03/03/2022].

[7] Herbsleb, J.D., and Mockus, A., (2003) *An empirical study of speed and communication in globally distributed software development·* IEEE Transactions on Software Engineering, Volume 29, Issue 6, p. 481-494. Available at:https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1205177 [Accessed: 03/03/2022].

[8] Kramer, M., (2018) *Best Practices in Systems Development Life Cycle: an analysis based on the waterfall model*. Review of Business and Finance Studies, Volume 9 , Issue 1, pp. 77-84. Available at: https://www.theibfr.com/download/rbfs/rbfs_2/rbfs-v9n1-2018/RBFS-V9N1-2018-6.pdf [Accessed: 02/03/2022].

[9] Lucassen, G., Dalpiaz, F., Martijin, J. E. M., Werf, VD., and Brinkkemper, S., (2016) *The Use and Effectiveness of User Stories in Practice.* International Working Conference on Requirements Engineering: Foundation for Software Quality. Available at: https://link.springer.com/chapter/10.1007/978-3-319-30282-9_14 [Accessed: 03/03/2022].

[10] Lucidchart (u.d) *UML Diagram Tool*. Available at: https://www.lucidchart.com/pages/examples/uml_diagram_tool [Accessed: 03/03/2022].

[11] Malik, R.S. Ahmad, S.S., and Hussain, M.T.H., (2019) *A Review Of Agile Methodology in IT Projects.* 2nd International Conference on Advanced Computing and Software Engineering (ICACSE - 2019). Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3351064 [Accessed: 03/03/2022].

[12] Markovic, M.R., and Salamzadeh, A, (2018) *The Importance of Communication in Business Management.* Available at: https://www.researchgate.net/publication/328630849_The_Importance_of_Communication_in_Business_Management [Accessed: 03/03/2022].

[13] Mayo Clinic (2019) Colour Blindness. Available at: https://www.mayoclinic.org/diseases-conditions/poor-color-vision/symptoms-causes/syc-20354988#:~:text=Color%20blindness%20is%20usually%20inherited,shades%20of%20blue%20and%20yellow [Accessed: 04/03/2022]

[14] McCormick, M., (2012) *Waterfall vs Agile Methodology*. Available at: http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf [Accessed: 02/03/2022].

[15] National Health Service (NHS) (2012) *Blindness and vision loss*. Available at: https://www.nhs.uk/conditions/vision-loss/ [Accessed: 03/3/02].

[16] National Health Service (NHS) (2019) *Colour Vision Deficiency* (colour Blindness). Available at: https://www.nhs.uk/conditions/colour-vision-deficiency/ [Accessed: 03/03/2022].

[17] Oza, N., Fagerholm, F., and Munch, J., (2013) *How does Kanban impact communication and collaboration in software engineering teams?* . 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 125-128. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6614747 [Accessed: 03/03/2022].

[18] Poret, S., Dony, R.D., and Gregori, S., (2009) *Image Processing for Colour blindness correction.* IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH), pp. 539-544. Available at:https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5444442 [Accessed: 03/03/202].

[19] Srivastava, A., Bhardwaj, S., and Sararswat, S., (2017) *SCRUM Model for agile methodology*. International conference on Computing, Communication and Automation ICCCA), pp. 864-869. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8229928 [Accessed: 03/03/2022].

[20] Sutherland, J., Viktorov, A., Blout, J., and Puntikov, N., (2007) *Distributed Scrum: Agile Project Management with Outsourced Development Teams.* 2007 40th Hawaii International conference on System Sciences (HICSS'07). Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4076936&tag=1 [Accessed: 03/03/022].

[21] World Health Organisation (WHO) (2021) *Blindness and Visual Impairment.* Available at: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment [Accessed 03/03/2022].

BIBLIOGRAPHY

[22] FlatLaf (2019) *FlatLaf - Flat look and Feel*. Available at: https://www.formdev.com/flatlaf/ [Accessed: 0/03/2022].

[23] Freetts (u.d) Available at: https://freetts.com/ [Accessed: 28/02/2022].

[24] GitHub (u.d) JLayer. Available at: https://github.com/wkpark/JLayer [Accessed: 04/03/2022].

[25] GitHub (u.d) Media Player. Available at: https://github.com/connorssweet/Media-Player [Accessed: 04/03/2022].

[26] Java2s (u.d) Setting the default button: JRootPane, Swing. Available at:http://www.java2s.com/Tutorial/Java/0240__Swing/Settingthedefaultbutton.htm [Accessed: 04/03/2022].

[27] Oracle (u.d) Class File. Available at: https://docs.oracle.com/javase/8/docs/api/java/io/File.html [Accessed: 04/03/2022].

[28] Oracle (u.d) File InputStream. Available at:https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html [Accessed: 04/03/2022].