COMP8860: SOFTWARE ENGINEERING

ASSIGNMENT 1 – GROUP PROJECT

GROUP 1

BIPIN RAI br322@kent.ac.uk

SAMEER GURUNG ssg20@kent.ac.uk

BRETT EVANS ble6@kent.ac.uk

ALEXANDER NIXON apjn2@kent.ac.uk

Introduction

Our project was focused on enhancing and incorporating accessibility features for students with visual disabilities into HEAT, a program designed for learning the Haskell language. To understand what the necessary requirements were, we engaged with our customers and conducted a comprehensive analysis of the current interface of HEAT where we identified key challenges faced by users with various visual impairments. These challenges included issues related to colour blindness, partial or complete blindness and visual reading difficulties like dyslexia.

We then decided to conduct research into the different types of visual impairments, with a focus on the most common types such as colour blindness, dyslexia and partial blindness. We explored how accessibility features, such as screen magnification, text-to-speech, colour settings, removal of visual clutter and higher visibility within the interface (notifying users of new features, as seen in the splash screen) could address these accessibility concerns users have.

Following this research, we engaged with our customers again (Sergey and Rogerio) to identify acceptable requirements. These included implementation of features such as text-to-speech for blind users, contrast options for colourblind users, implementation of a splash screen, ability to adjust font sizes and new icons for partially blind users. Due to the projects time constraints, our team had to discuss what was realistic to implement within our one week time frame whilst also ensuring alignment with the customer requirements.

The approach we decided to implement throughout this project was the use of the Agile development methodology. We chose Agile as it emphasizes small and frequent improvements to our functions and incorporation of more functions if there was enough time.

Group Organisation

Our team worked within the framework of Agile methodology, where initiated the task delegation process by engaging with users and formulating user stories and requirements. Through user dialogue, we prioritized features based on user feedback and their perceived importance for software implementation, we then assessed the feasibility and difficulty of implementation within the limited time frame and created tasks based on their difficulty and importance for implementation. Tasks were subsequently assigned to individual developers, with easier tasks taking precedence, except for Alex who undertook implementation of Text-To-Speech which was categorized as a challenging task.
Originally, our initial plan involved developers transitioning to another user story after completing their initial task, with the intention to add more functionality to improve the accessibility to HEAT. However, a mid-week review including a customer walkthrough with Sergey and discussions with Rogerio provided feedback with our initial functions. This feedback prompted us to make a decision to abandon less prioritized functions which were also harder to implement, such as dark mode and colour-changing text, redirecting our focus toward refining the core functionalities which were defined through our initial discussion with customers.

Through the time constraints and guidance by customer feedback, our team ultimately concentrated on the essential functions prioritizing both importance and the ease of implementation. This strategic shift in group organization ensured that our efforts were realistically aligned with the project’s time frame and the overarching goals of enhancing the accessibility for customers with visual impairments.

Development Process

Through our development process, we attempted to adhere to the Agile principles and follow the guidelines for the Scrum methodology where possible. The typical steps of Scrum are:

1. Sprint planning
2. Daily standups
3. Sprint review
4. Sprint retrospective
5. Backlog refinement

Using these guidelines, we managed to implement all these steps in some capacity (Refer to Wiki for details). Deviation to this methodology was due to inexperience resulting in difficulties to fully engage with typical Scrum ceremonies. For example, our sprint planning had to be conducted during the actual sprint as our initial user stories did not involve the customers but rather just ideas from the developers. Therefore revisions to user stories were made with customer involvement. Another deviation we had from the typical Scrum process was during our daily standups. Typically these are 15 minutes, however in our case these always overran the scheduled timeframe due to discussions relating to code or revisions to our initial plans due to customer feedback.

Furthermore, the time constraints of this project meant that we had to customize our Sprint to less than a week rather than a typical Sprint that lasts usually lasts two to four weeks. This deviation was to simplify the Scrum processes by focusing on essential ceremonies to ensure we can implement, improve and deliver working functionalities within our given time and apply the Agile principles.

Customers were involved throughout all stages of the project (Refer to diary in Wiki) as in our initial interface walkthrough Sergey was present and we collectively identified major interface issues and what functions could be implemented. Rogerio also discussed further functions we could implement to improve accessibility within the interface. After our initial walkthrough, Sergey was consulted again to refine our user stories to ensure clarity on the requirements and expectations regarding functions. Feedback was again provided during our review where Sergey acted as a customer to simulate different user stories in which we could test the functions that were implemented within the interface. This allowed us to follow Agile principles where we could improve our requirements through iterative development and functional testing ensuring alignment with customer requirements. This continuous involvement with customers facilitated collaboration between the developers and ensuring the final product met the needs of the customers.

The distribution of tasks between developers were based on how difficult the task was and their priority in ensuring a accessible interface for the user. As a collective we decided we should focus on the easier tasks that were a priority for a functional final product such as increasing the font sizes, changing the icons and a splash screen (Refer to group organisation within Wiki). However, we decided that Alex would be assigned with a difficult task so that if he was successful we would have implemented a feature for users who are fully blind as well, improving the accessibility of HEAT. To ensure everyone was satisfied with the tasks assigned to them the other 3 developers would try to work on tasks if possible throughout the week whilst Alex would solely focus in implementing text-to-speech. We also had stand-ups to address any concerns between the team and ensure the workload delegated to each developer was manageable and completable.

Requirements (Solutions)

We had several approaches in mind, particularly focusing on improving accessibility for the visually impaired. Upon launching HEAT, we identified a few issues which was confirmed with Sergey during a customer walkthrough. Firstly, the interface was not visually appealing, even for users without visual impairments. The icons and font size were excessively small, causing discomfort even for the average user. Additionally, the icons, especially the help icons, were challenging to distinguish. Furthermore, the absence of text-to-speech features rendered the program unusable for totally blind individuals and challenging for those with partial blindness. The lack of a splash screen also contributed to decreased accessibility. Before addressing these issues, preliminary research was necessary. To implement text-to-speech functionality, we had to search for a suitable library, ultimately choosing freeTTS. Changing icons required research and the use of online tools, such as an image resizer and a royalty-free library of icons. Adjusting font size and adding splash screens involved less of internet research and more studying the HEAT code, to understand how improvements could be made.

We also had to consider the customers' (Sergey and Rogério) preferences. In light of this we designed user stories that aided us in evaluating their wishes. Additionally, establishing an acceptance criteria guided us in understanding when the product would be satisfactory. Regular conversations with clients were essential in aligning our work with their vision. During discussions about text-to-speech, a crucial insight was made by Rogério- a blind person might struggle to locate the icon to activate the text-to-speech. This led us to approach implementing hotkeys, enhancing the product's accessibility and aligning it more closely with the customer's vision.

Considering these factors, we opted to assign specific tasks to each team member, dividing responsibilities into four major roles. These included implementing fundamental text-to-speech functionality, creating a splash screen to showcase new features and options, enhancing font customisation with sliders, and increasing the size of icons. At first we decided to tackle all approaches in our respective areas, attempting to implement as many of the user stories as possible. We prioritised certain tasks, giving difficulty levels from low-high, which required us to research into the potential difficulty of each job. Despite this some approaches were still too unimportant or too impractical to be implemented in the given time frame. An example of this was screen magnification, we considered this too unimportant, especially since the customers did not explicitly request such a feature; instead we focused on the more difficult and important features. We still attempted to do as much as possible, so implementations for dark mode, button for swapping from old to new icons, colour blindness etc were all attempted, however we had the hindsight to consider that these were of a low priority and were too difficult to be completed in the timeframe. Although they were requested by the user, such as the "shut up" button, to terminate talk-to-speech mid speech, we had to consider our options mid development, and this became a low priority.

We did an initial walkthrough of the interface with Sergey. We went through a number of studies such as for types of colour blindness and dyslexics, so that we could cater to the needs of a larger demographic of people. All the research that we did can be found on the wiki under the requirements, background study section. Everything is documented extensively on the wiki, including what should be modified in HEAT. We also had our own word documentation for documentation, and compiled everything collectively at the end.

Design

We designed the features based on the requirements from the ranking of user stories, the ones that were deemed easy to implement such as icon changes, a splash screen and adjusting font sizes. One of the challenging requirements was text-to-speech, which was assigned to Alex who worked on it throughout the week. Additional features such as dark mode, changing text colours were initially planned for later implementation once initial features were completed however these were scrapped as they were too complex post mid-week scrum.

The design ideas of the new interface functions such as splash screen, buttons and sliders were accomplished using Figma where we developed lo-fi prototypes to present to users. Figma provided an easy to use tool to build and manipulate new features as creating them within Java would be more challenging for clarity. Icon testers were manually created within Word and they are imported from a base folder which follows our initial research intentions. These lo-fi prototypes were further developed through feedback from Sergey and we attempted to implement in our final product.

The design documentation was developed concurrently whilst we coded the functions. This approach allowed us to get a basic understanding of what functions need implementations and what design choices need improvements. However, in hindsight we should've conducted a more detailed pre-development planning phase which would have made designing and implementing our ideas easier.

These design features were documented through various steps:

1. **User stories and requirements**: Initial documentation was when we created user stories and their requirements, this formed the foundation for understanding the functionality and design needs.
2. **Lofi prototypes in Figma** – To visualize and iterate our design concepts, we created lo-fi prototypes within Figma, these were used to represent our initial ideas to Sergey who provided feedback on iterating the design to meet the customer requirements.
3. **Improvements during implementation**: As we started programming the functions we kept the design section within the Wiki up to date with our new designs. This page has a comprehensive overview of the design choices we made throughout the implementation process.

Implementation and Code Quality

Research proved useful for helping to determine whether a feature would be easy or obvious. For certain approaches, some methods were very clear-cut obvious, with not a lot of other ways to implement them. For example, for text-to-speech it was obvious that we needed to download a library and import it into the Eclipse environment, in order to achieve any functionality. Additionally, it was obvious we needed a method (performTextToSpeech) to do text-to-speech, taking a String as a parameter. However, it was not entirely clear how reading the console would actually work, thus numerous approaches had to be taken, such as trying as many libraries as possible, starting with a scanner and ultimately ending with an element iterator, so it would play out on a different thread.

For the SplashScreen, this appeared to be a fairly obvious process when starting, as we could just use a lot of the pre-existing code to make a JFrame, that appears with options and updates. However, at the end we discussed how it may have been better to use a JPanel instead, which is superior for combining group-related components together. In terms of adding buttons, this was fairly obvious and a lot of features were taken from the pre-existing code. Icons was also fairly straightforward, as we knew we needed a new folder to hold the larger icons in. However, when we tried to implement a toggle button that changes between the smaller and larger icons, we could not get it to work. However, after speaking with Sergey, he was content with just having the larger icons and no button, so this was fine.

Locating the relevant parts of the HEAT code for modification was primarily a trial-and-error process. This was partially a result of the sparseness of the documentation. Occasionally, there were scarce one-line comments offering brief insights into the purpose of classes or methods, but overall clarity was lacking. To identify relevant sections, we often traced back from the Main class, exploring the classes and methods it invoked to initialise the program. For instance, following the path from the Main class to the createGUI method in the window manager. This is where Eclipse proved to be a very useful IDE, as it provided a way to jump directly to the library/class that was being used by another, helping speed up the process.

In terms of issues we encountered some problems with its general functionality. The program exhibited occasional bugs, such as instances where clicking on the upper console would intermittently lock users out of typing in the lower console. We tried to make this a smoother experience for the user as much as possible, but given time constraints it was difficult to implement new features while fixing bugs of old features, especially with consideration to the lack of documentation on HEAT. However, we did take necessary measures when we could, such as stopping the screen from freezing when TTS was active, as requested by Sergey.

Regarding our own features, we ran into some issues with implementation, especially with the features we considered to be of high difficulty. One of these was settings for colour blindness. At first we attempted to implement a way to change the text on the screen all to black, before making colour specific settings, however we were not able to achieve this in the given timeframe given our other tasks. Another was dark mode, considering we were not able to change the colour the text, we were also not able to change the colour of the backdrop in the given time as well, unfortunately this meant we had to push this to the bottom of the list until we had approached the other tasks. Additionally, we tried to implement a quiet button into the text to speech functionality, however we ran out of time to give it functionality, and the button remains an empty husk on the GUI.

We can use the acceptance criteria from the user stories to decide if a feature has been "successfully" implemented. The acceptance criteria for text-to-speech related user stories was, Implement text to speech, include an easily accessible button on the keyboard and GUI to read out the code and output to the user. This is mostly a successful implementation, instead of one button we have two buttons, one that reads out the entire text and another for reading out the current line the user is on. The splash screen was another feature successfully implemented, the user wanted a splashscreen to appear including new features and updates. Although, Rogério originally requested that all the options be accessible on the splashscreen, due to the difficulty of this, we instead talked with Sergey who informed us he would like a settings tab for the options appear alongside the splashscreen, which functionally works very similar to having a splashscreen with all the options. Furthermore, there are now more accessible buttons for the font size, rather than a long list of numbers, we have implemented successfully buttons for small, medium and large and made the base font size of the HEAT application bigger. Finally, new, bigger and clearer images for icons were also successfully implemented, and are roughly 30% bigger than the original; Sergey viewed this and made a comment about how much nicer the new icons looked, fulfilling our acceptance criteria.

Quality Assurance

The testing procedures were drawn up to align with the nature of the features we implemented, mainly focused on testing the changes we made to the interface. Collectively we devised a comprehensive testing procedure incorporating integration testing, acceptance testing, code reviews and usability testing (Refer to Wiki). Given the nature of the implementation and changes we made to the code, we opted to not implement unit testing due to concerns about the time spending setting up the infrastructure to conduct unit testing outweighing the benefits gained from just functional testing alone.

The feedback we obtained from involving customers through acceptance testing ensured that our functions to the interface were aligned to their expectations. There were various difficulties in conducting integration testing as we encountered issues in merging due to the differences between the main java file that Brett and Alex were working on. This in turn led to isolate the issues and we had to revert some merges within GitLab.

As development was mainly done in person, we used pair programming, especially between Bipin and Sameer as they were implementing similar functionalities. They were able to regularly use code inspections and review each-others functions. For example, Sameer was having issues getting the apply button to function properly so they worked together and found the line of code causing the issue and managed to successfully fix the issue. Another example was when Bipin had issues implementing functional buttons within the splash screen, through code inspections we figured out he was using a different library compared to the new settings tab. However, due to time constraints we could not fix this issue and had to scrap a function.

Overall, we have great confidence in the effectiveness of our changes due to the various testing procedures that were established. We conducted thorough testing internally and actively involved Sergey to ensure the new software interface addressed the user stories (Refer to Wiki). Using this iterative approach to testing and refinement, it was possible for us to deliver a functional final product which addresses the main issues highlighted by the developers and customers.

Tools

Throughout the week, our group used GitLab as the repository for our project. Each developer created different branches from the main file on GitLab to develop and test code for their tasked functionality. We utilized GitLab's issue tracking tool to monitor progress on the challenges we encountered during functional development.

This included reassigning issues among developers and updating issues accordingly so we can bring it up during our daily stand-ups meetings. GitLab also was the platform for our Wiki, where we documented the project details such as the development process, initial design ideas, user stories and requirements needed. The Wiki also has some code snippets, utilizing GitLab's embedded tool to show lines of code that were either implemented or scrapped from the final product. Additionally, we used GitLab to review other developers codes through merge requests, allowing for collaborative evaluations and approval before merging our branches into the main repository.

Apart from GitLab, our team also utilized the following tools which are all referenced within our Wiki:

- **Figma:** We used Figma to design our initial lo-fi prototypes, which were presented to Sergey for feedback on our initial design concepts and adjusted accordingly to meet customer requirements.
- **Eclipse IDE:** The entire group utilised Eclipse as our integrated development environment, which allowed us to import the repository from Git, export our project as JAR files, push and pull each other's branches with GitLab.
- **Generative AI (ChatGPT 3.5):** ChatGPT was utilized to enhance our code syntax accuracy such as why a line of code might not be working or what we needed to add to have a functioning program.
- **FreeTTS API:** This was successfully implemented into our application to have a working Text-To-Speech function.
- **Icon Resizer tool and Royalty-free Icon site**: We used these to find images that were relevant to our project, resized them into different dimensions and tested their compatibility with the HEAT application.

Understanding & insight

In regards to the modifications we made to HEAT, we are satisfied with the main functions that we managed to implement.

Our main functions that have been implemented such as Text-to-speech (TTS), we realized post implementation we could have consulted more with Sergey during the design process which would have allowed us to incorporate a hotkey functionality earlier, bypassing the needs for the buttons in the hot bar as blind users won't be able to see the icons. While the functionality works, the code quality for

TTS was a bit messy, especially as the code style diverges from the rest of the program such as the main menu and the toolbar. Additionally, the stop button for TTS needs refinement as it has no functionality.

For the icons feature that was implemented, we are content with the changes made and in alignment with Sergey’s expectations as he wanted new icons that are larger and easier to understand their meaning. However, in terms of the code quality there were challenges when merging, as it required direct code copying from the branch to the main file. Redundant parts from previous attempts to get a button to change button sizes also exist within the code but it is not functional.

The splash screen while functional based on initial design feedback from Rogerio did not satisfy feedback from Sergey; Sergey wanted a static window informing users about the updates. However, the settings tab is successful in terms of its functionality and how we were able to successfully customize it by removing the old Font Size combo boxes (Refer to Wiki). We acknowledge that we fell short of expectations as the settings tab is empty and should have in retrospect had additional features such as background colour and changing text colours. The code quality for the settings tab is generally good as its functioning as desired but it could have benefitted from more comments for clarity.

Finally the Font size feature is satisfactory at best as we realize we missed the opportunity to implement extra features such as choosing font styles and font choices. Also we could have consulted with more customers throughout the week in regards to button size preferences as the size of the pre-set buttons were created without consultation. In regards to the functions, with more time we could have refined aspects of TTS, Icons, Splash screen, Settings tab and the font size features, addressing code quality whilst also incorporating additional functionalities based on the user stories we had.

As well as this, we personally feel that more could have done in the week prior with regards to planning and research so that development could have been started sooner. We feel that we did not maybe use GitLab to its fullest potential which caused issues during integration testing when integrating Brett’s functions into the master branch. As well as this, we also feel that we maybe could have explored the code base more so that methods or codes which needed to be changed could have been identified sooner. Overall, a lot of management issues caused us to lose time that could have been used to involve customers during our developmental processes and perhaps further refine our end product. But, overall as a group we are satisfied with what we have done, we feel that we did what we could have done given our limited experience and time constraints we were under and took this week in stride as a learning experience in how to effectively work in a software engineering environment. Having exposed ourselves to engaging with stakeholders and using GitLab for collaborative development; We can confidently say that we could produce an even better result in the future.

Conclusion

In conclusion, as a group we are content and satisfied with the features we managed to successfully implement to enhance user accessibility in our project. The functions that were implemented such as Text-to-speech, redesigned icons, settings tab customization and the splash screen shows that we managed to generally address our customer’s needs. However, we have to acknowledge that this project functionality could have been improved had there been more time.