**COMP8860 Group 5 - Project Report**

**Disclaimer: JAR FILE DOES NOT INCLUDE TTS FEATURE BUT RUNNING FROM MAIN DOES**

**Introduction**

The group was tasked with creating additional features for the Haskell IDE, Heat, with a particular focus on features that would aid visual impairment. Project information was released 1-week prior to a dedicated 1-week sprint. The work was completed in a group of 4 with the deliverable including a splash screen, enlarged icons/text, a magnifying glass and a text to speech feature in the editor.

**Group**

Luis Goate – Developer

Manish Tripathi – Developer

Nishan Patel – Developer

Thushanthan Shanthakumar– Developer

**Development Process and Requirements**

* Due to the short nature of the project and requirement to regularly engage the customers, the agile methodology was adopted in order to cater towards the project’s needs.
* This included daily in-person standups in the morning, sprint planning sessions, regular interaction with the customer and working in a cross functional team whereby all members were expected to contribute.
* The customer was engaged throughout the process. Once before the sprint week began in a scoping email sent to ensure we were on the right track in our planning, and then daily throughout the week to ask for feedback on feature process. Key insights were to explain which group of visually impaired users did each feature cater towards, adding a shortcut key to the text to speech (if possible) and the importance of the splash screen to the overall deliverable.
* Work was distributed into preliminary tasks anA white sheet with blue text

  Description automatically generatedd development work.

Figure 1 Preliminary Tasks

* Before the sprint, we set out a set of preliminary tasks that needed to be completed to ensure a smooth transition into the sprint.
* Development work was then split depending on appetite for a challenge with a focus on ensuring each member feeling they were able to contribute.

Figure 2 Gant chart

A screenshot of a computer

Description automatically generated

* Above depicts the Gant chart of our epics devised in the planning features.
* A screenshot of a computer

  Description automatically generatedBelow shows the user stories and tasks in a kanban board that were referred to throughout the development process.

Figure 3 Kanban Board

* Both views are created on our agile project management tool JIRA.
* A group decision was made whereby we decided to reduce the depth of the features to focus on breadth. Team members then rotated once their feature had been finished to assist other members features and version control.

**Design**

* Broadly speaking, making things bigger, allowing users to hear things, making accessibility options accessible (splash screen) and colour contrast options ticks the boxes for several visual impairment conditions (WHO, 2025).
* From requirements to the design, development followed a pattern of a series of proof of concepts and engagement with the group and customers. Once we achieved progress, we then assigned context to our features. This way, any research would be tightly coupled to the progress of the actual feature and we wouldn’t oversell the deliverable.
* Documentation has been limited to the user stories and tasks shown in figure 3.

**Implementation and Code Quality**

* We successfully implemented the following features:

**TTS (Editor mode)** *Luis*

* Implementation took 1-2 days and further regression testing took a few hours.
* ChatGPT was used to formulate the base code that provided the POC – a GUI text editor with highlight TTS.
* Implementation of the button and integrating the code was done manually and this task made-up most of the work.
* Generally speaking, comments were implemented in the new code with the majority of the new code being traceable through ‘Action Manager’.

A screenshot of a computer program

Description automatically generated

Figure 4 Git history for feature TTS

**Large Icons and Text** *Manish*

* The implementation process took **2-3 days**, including design, coding, and integration.
* The feature was introduced to **enhance accessibility**, ensuring better visibility for users with visual impairments.
* **ChatGPT** was used to generate base code for resizing UI elements, while **manual adjustments** were made for optimal usability.
* The majority of the code changes were made in **WindowManager**, **SettingsManager**, and **ActionManager**, with some updates in GUI-related files.
* Comments were added to **improve maintainability** and allow for future enhancements.

**Magnifying glass** *Nishan*

* Implemented on day 3
* Feature allows user to view a magnified version on any part of the screen (PC/MAC)
* Code stems from action manager

**Splash screen** *Thushantan*

* Implemented on day 5
* Feature includes loading bar, background visuals and buttons to visual accessibility mode
* Code stems

**Quality Assurance**

* Regression testing throughout the development process was performed
* The tests included:
* Ensuring the buttons worked after the text size changes had been implemented.
* Ensuring the text to speech feature worked and that subsequent features such as copy and paste also worked.
* Ensuring the magnifying glass worked and that subsequent features such as copy and paste also worked.
* Testing splash screen

**Tools**

* Gitlab was used for version control. Each member made at least one commit. CI/CD was beyond the scope of the project. We spent a significant amount of time integrating our code and familiarizing ourselves with version control.
* Atlassian JIRA, was used as our project management tool. This allowed us to produce Gant charts/Epics/User Stories/Tasks. This also acted as our accountability tool and source of truth regarding the work we set out to achieve.
* IntelliJ, the Java IDE, was chosen in order to development due to the group’s familiarity with the tool.

**Understanding & insight**

* A previously mentioned, most visual impairments could be catered toward by applying the 4 features we set out to achieve. In terms of successful completion of the features, it’s fair to say we made inroads however further integration development would be required to make the features cohesive and comprehensive. For example, the text to speech feature works in the editor however it does not reach the toolbar or menu options. Furthermore, the splash screen does not provide settings for the visually impaired user to choose from meaning the user may be unaware of what’s available – instead we offer a combined ‘Visually impaired mode’.
* ChatGPT was useful for all members with each member using it to varying degrees. ChatGPT allows the feature to upload the project completely and ask for recommendations, which, on the surface seems like a solution to adding features. However, often times, during debugging, the code produced by GPT was beyond the skill level of the developers and hence debugging took up a significant amount of time as it was tricky to solve problems from first principles. Used for small sections of code however ChatGPT proved to be an effective tool.
* In terms of additional setbacks, Luis encountered a malfunctioning laptop on day one. Thushanthan also experienced unique IDE related issued which slowed down development time. Manish had to pivot the to a different feature when building the high contrast feature on day 3.
* In terms of what went well, daily standups and helping each other out worked quite well. Each team member was willing to assist each other.

**Conclusions**

* Good learning experience, which gave us a taste of what to expect in the real world whilst working toward a deadline. Next time we would spend more time in the planning stage and scoping out the work better. This would allow us to be more precise in our feature creation and aid with division of labour so that members could have worked with their strengths.