**Final Report - Title of the software project**

*List the names of all team members*

1. **Implementation Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Implementation Plan** | **Progress** | **Comments** |
| In-001 | The user will input the amount of time they want to spend on this task. This task can be writing or composing music. | 100% | When the user fills the form for the commitment, the data will be serialized in a .Tasker folder and stored within it, a .taskerdata file. |
| In-002 | The user will select what type of task they want to time themselves doing—writing, music or drawing. | 100% | When the user fills the form for the commitment, the data will be serialized in a .Tasker folder and stored within it, a .taskerdata file. |
| InK-003 | Temporarily buffer the user’s interaction with keyboard for analysis. | 100% | Only runs on Linux - macOS will require a unique implementation, likely written in Swift or Objective-C. Linux has a <input.h> header file, that gives the KeyboardListener what it needs to capture keystrokes. There is no such header file on macOS, as far as we were able to discern. |
| Out-004 | Show how much time has been spent on the task (writing or composing music) compared to  how much time the user has spent on this session. | 0% | There exists backend code to calculate the delta between total time and productive time, as well as unproductive time, but the program is not utilizing this code yet. (it will, soon) |
| InA-005 | Temporarily buffer the user’s interaction with Audio/Microphone for analysis. | 100% | In order to reduce CPU usage, we must sleep the current thread for at least 2 seconds for each iteration that AudioListener is looking for an audio signal. Without sleeping, CPU usage will skyrocket. |
| In-006 | Timed Task(s) will be saved for future analysis. | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| Ink-007 | Shortcut to make Tasker appear in front of user—maximize Tasker window. | 0% | Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-008 | Show the current state of Tasker’s timer. | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| Out-009 | Show the current progress towards the set goal in a user-friendly manner. For example, “You have completed 50% of your task.” | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| Out-010 | Show the current progress for a specific task(writing, composing music or drawing) for the past week. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-011 | Show the current progress for a specific task(writing or composing music) for the past month. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-012 | Show the current progress for a specific task(writing or composing music,) for the past year. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-013 | Plot in graphs the current progress for a specific task(writing, composing music, or drawing) for the past year. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-014 | Plot in graphs the current progress for a specific task(writing or composing music) for the past month. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-015 | Plot in graphs the current progress for a specific task(writing or composing music) for the past week. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-016 | Plot into graphs Milestones. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| Out-017 | After two weeks of analyzing a task, Tasker can start personalizing the sensitivity of the timer. | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| PR-018 | Proactively clear data that is not needed to ensure we take care of the user’s privacy when interacting with keyboard. | 100% | No keystrokes are buffered at any moment during runtime. |
| PR-019 | Notify the user when they are done with their task. This means when productive time has reached the goal. | 0% | In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application. |
| PR-020 | Proactively clear data that is not needed to ensure we take care of the user’s privacy when interacting with microphone. | 100% | No audio signals are recorded to disk at any moment during runtime. |
| In-021 | The user will be able to create a commitment to play music, write or draw for an amount of time they’ll specify. Such as from 1 hour/day to 4 times/week. | 100% | When a commitment is created, the user will be able to parameterize the interval of time spent on a task for each session. |
| In-022 | Tasker’s window will hide from screen when the timer starts. | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| In-023 | The user will be able to view their commitments. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| In-024 | The user may choose how to be notified at the end of their session. | 0% | **In order for this to be implemented, Timer must be implemented. We were not able to work on this portion of the application.** |
| Out-025 | The user will be able check to their progress on a per-session basis. | 0% | CommStatsQWidget, the class responsible for this functionality, has not been fully implemented. Our efforts were diverted to other portions of the application that required our attention, mainly AudioListener and the Timer — particularly with respect to management of threading. |
| PR-026 | Calculate the amount of keyboard activity during session for use in characters per minute. | 0% | **This feature has been omitted for our SRS.** |
| Input-027 | Ask for the user’s consent before accessing any hardware peripheral—keyboard or microphone. | 0% | macOS will ask the user before granting Tasker permission to use the microphone. Therefore, we did not need to implement this feature. |
| PR-028 | Have an algorithm that accurately counts down the timer as the user interacts with keyboard—as they write. | 0% |  |
| In-029 | Allow the user to turn off consent for any hardware peripheral—keyboard or microphone. | 0% | macOS will ask the user before granting Tasker permission to use the microphone. Therefore, we did not need to implement this feature. |
| PR-030 | Our application will only run on unix-based desktop. | 100% | Tasker has been tested on macOS (10.15 Catalina) and Linux Mint (Tina), which are UNIX-based operating systems. |

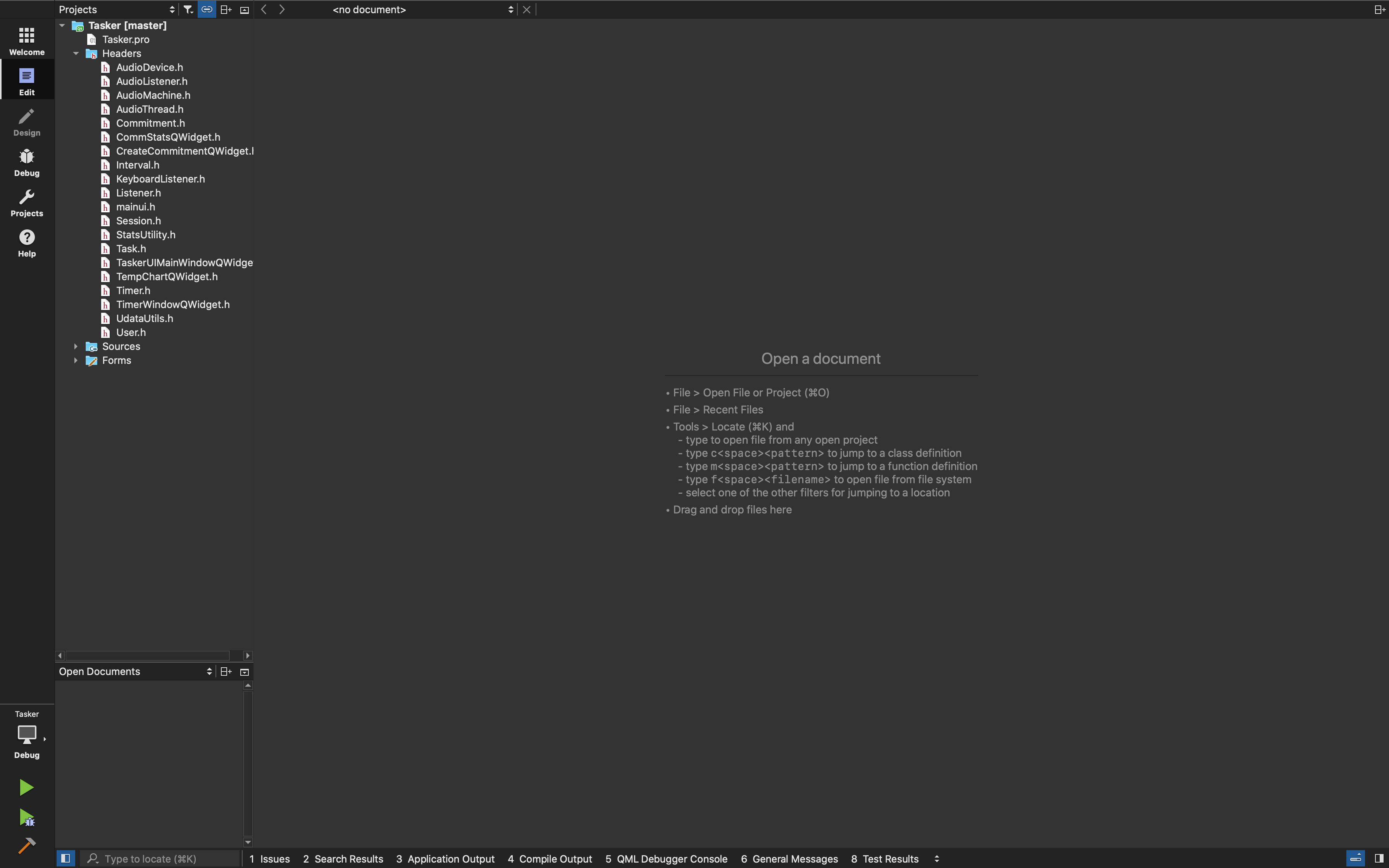
1. **Building and Running Instruction**

Decompress the archive file, Tasker\_Alpha.zip.

Upon decompressing the archive file, the following directory, Tasker, will appear in the path where you decompressed the archive.

Next, you will launch Qt Creator (you must have at least Qt 5.13.2 or newer to build and run Tasker) — click Configure Project if you are prompted with a menu and drop-down window describing the compiler type. (it will default to clang on macOS, or GCC on a Linux system)

Next, you will hit the ‘play’ button icon to build and run Tasker.

(bottom left of Qt Creator)

1. **Bug Report**

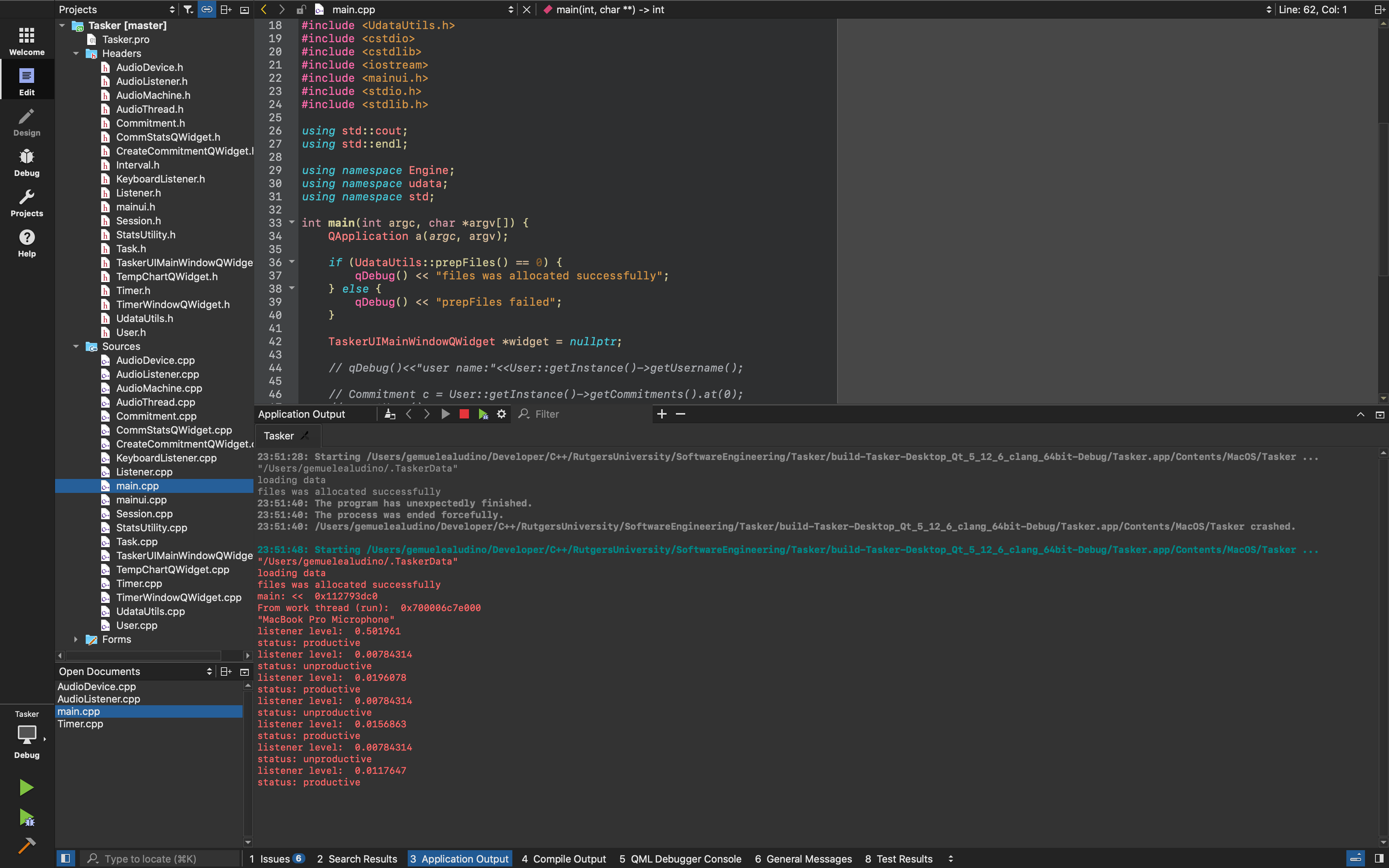
We had reused Qt’s Audio Input Example project as a baseline,

and removed their GUI portion — as all we were seeking was the ability to sense audio signals from the chosen input device. (As of now, Tasker defaults to the input device assigned by the system — if a user wants to change their input device, they will have to do it in their system preferences. We plan to rectify this as soon as the GUI takes shape)

The input sensitivity, when set to 0%, produces a different value when AudioListener reports the signal value back, depending on the microphone type. We tested with the MacBook Pro’s internal microphone, as well as an Audiobox USB sound card, which accepts external microphone input.

The workaround, which was really more of a “hack”, was to “profile” what signal value would be returned when a 0% sensitivity value was set for a given microphone. Then, we would subtract the profile value from each AudioListener capture. That way, “0” really means “0”, and not some arbitrary value returned by the microphone when the input sensitivity is turned all the way down.

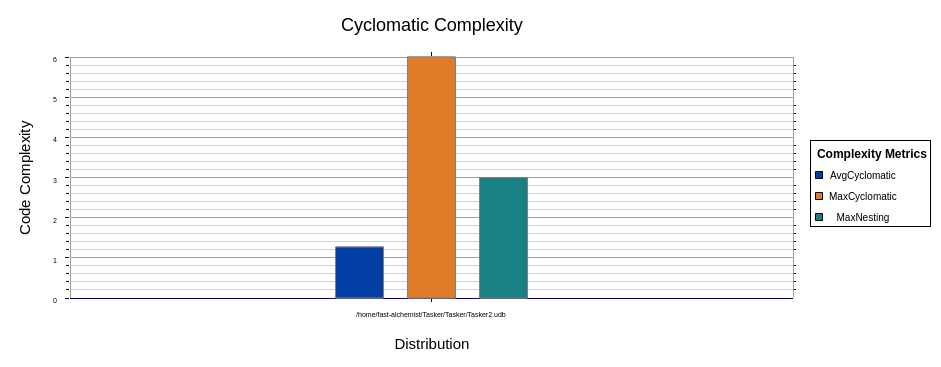
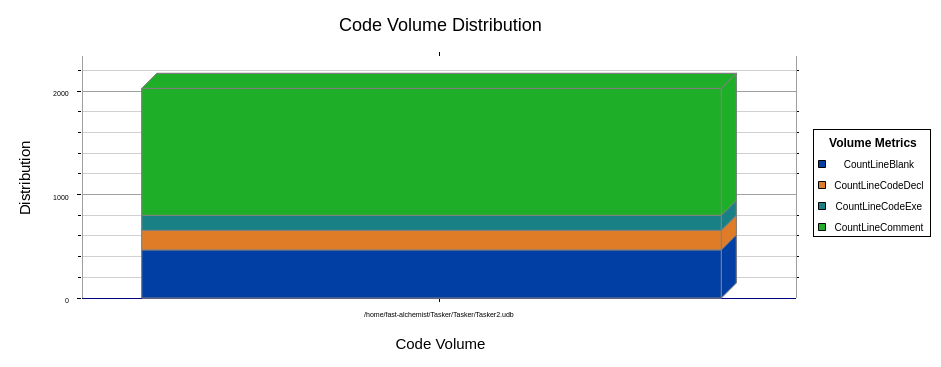
We must note that the sensitivity value returned has no connection to dB (decibels), we are simply interpreting this data for our use internally, say if the user wants to set a lower/higher sensitivity for sensing productivity, or lack thereof.

1. **Screen Shot**

For now, we are able to demonstrate functionality of the AudioListener - it can sense if the user is **“productive”** (the signal is above a predefined threshold), or if the user is **“unproductive”** (the signal is below the predefined threshold).

As of now, the AudioListener is able to work on macOS (tested on 10.15 Catalina) and Linux Mint (Tina) — the KeyboardListener is only working on Linux Mint at this time.

Therefore, the most “complete” experience with Tasker will require building and running it on Linux. Having said that, Tasker will build and run on both macOS and Linux.

1. **Project Metrics**

*Project Tasker2.udb*

*Blank Lines 465*

*Classes 20*

*Code Lines 1,667*

*Comment Lines 1,227*

*Comment to Code Ratio 0.74*

*Declarative Statements 625*

*Executable Statements 149*

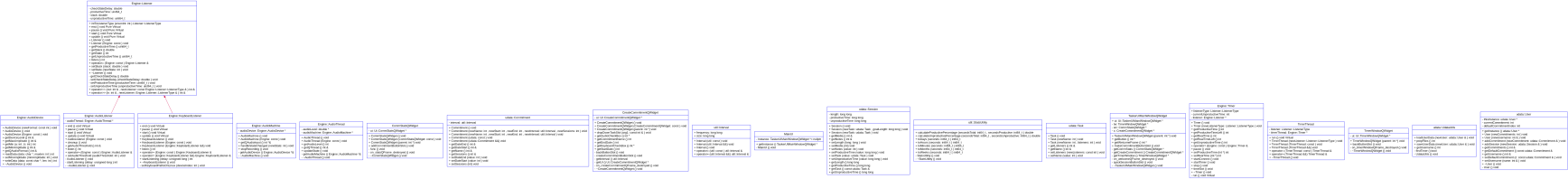
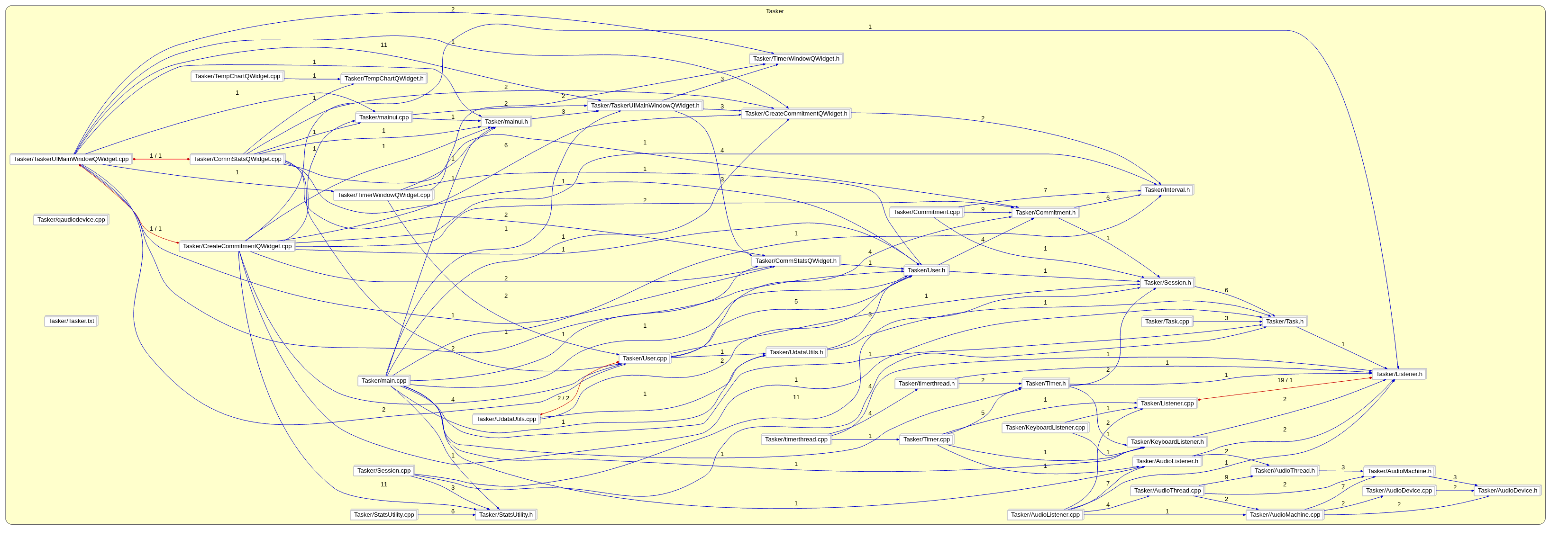
*Files 44*

*Functions 162*

*Inactive Lines 67*

*Lines 3,687*

*Preprocessor Lines 301*

**