1. **Team Name: Sleepy Ducks**
2. **Team Leader for this deliverable: Lorenzo Gomez**
3. **Team Members: Anton Ryjov, Gemuele Aludino, Lorenzo Gomez**
4. **Meetings:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Time-date** | **Attendees** | **Agenda** | **Action Items (who will do what)** |
| 11-04-2019  (Monday)  16:40 – 19:00 | Lorenzo, Anton,  Gem | Sketch core entity classes | Lorenzo  recognize entity classes(commitment, Task, Session, etc) and discuss their roles  Anton  recognize entity classes(commitment, Task, Session, etc) and discuss their roles  Gem  recognize entity classes(commitment, Task, Session, etc) and discuss their roles |
| 11-05-2019  (Tuesday)  18 – 20:00 | Lorenzo, Anton,  Gem | Sketch core utility classes | Lorenzo  recognize utility class and its role for presentation layer(stats)  Gem  recognize utility class and its role for presentation layer(stats)  Anton  recognize utility class and its role for  presentation layer(stats) |
| 11-06-2019 (Wednesday)  16:40 - 19:00 | Lorenzo, Anton,  Gem | Discuss classes that will persist on disk(user commitments data) | Lorenzo  Sketch data-persistent classes and variable naming convention for Commitment Window  Gem  Sketch data-persistent classes and variable naming convention for Stats Window  Anton  Sketch data-persistent classes and  variable naming convention for  TimerWindow |

1. **Weekly Time Logs:**

|  |  |  |
| --- | --- | --- |
| **Person** | **Total Time in minutes** | **Tasks** |
| Lorenzo | 600 | Work on CreateCommitment window |
| Anton | 600 | Work on Stats and commitments view |
| Gem | 600 | Work on Timer Window |
| **Total Time:** | 1800 |  |

1. **Issues:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue Number** | **Discovery Date** | **Resolution Date ( Est. – Act. )** | **Responsible Person** | **Description ( Prob / Resolution )** |
| Abstracting Listeners | 11-07-2019 | 11-11-2019 | Lorenzo,Anton,Gem | On our Thursday meeting, we realized that abstracting complexity is HARD. We want our listener code to be flexible enough, but also well-define for users to be able to interface easily with them. |

1. **Files and repository locations:**

|  |  |  |
| --- | --- | --- |
| **Filename** | **Location** | **Contents** |
| Tasker Class Diagram - Lorenzo.vpp.lc | https://github.com/thebigG/Tasker/tree/hardware-keyboard/documentation/visual\_paradigm/vpp | Class diagram |

1. **Plans for Coming Week:**

* Start implementing listeners
* Employ a good OO(usable and flexible abstractions) design into our implementation

1. **Comments:** *a paragraph from each engineer describing what they have done/learned from this deliverable*

**Engineer 1:** *Lorenzo Gomez*

*As I write this, my brain is fried. Abstracting entities is hard! Especially when those entities have coroutines baked into them that will be running in the background at all times. This assignment was really cool because it allowed us to architect our system and get an in-depth understanding of the static structure of our system. Even though we are designing static entities and abstract data types, thinking about these forced us to face the hard parts of our system that will be challenging to implement employing good OO practices. Since our system will be interacting with hardware a lot, it’s hard to abstract an interface that is 1) easy-to-use, 2) clean, 3)flexible. There are many challenges we overcame during this assignment and may doubts were resolved. However, there is still so much more to do(even in terms of redifining some abstractions)! Can’t wait to start implementing! With a clean and organized OO approach, of course!*

**Engineer 2:** *Gemuele (Gem) Aludino*

*This weeks focus was dialing down the classes we intend to use for Tasker — everything from fields, to member functions, and the relationships between classes. Seeing everything in plain writing helped to solidify our goals, driving home the point that planning is half the battle! In addition, we’ve agreed to naming conventions for all classes, member functions, and source files. This is something I’ve been wanting to push since the beginning, for I am very particular about this sort of thing (as we all should be). Regardless, I can’t wait to start implementing code already! In the interim, I will be brushing up on some C++ — despite Qt’s framework being a great help, its use still requires a strong understanding of idiomatic C++…we don’t want to make it obvious that we all came from a Java/C background. I want all of us to write good C++ code, and preferably, modern C++.*

**Engineer 3:** *Anton Ryjov*

*It was great to get a lot of the implementation onto paper and into our heads. One of the things we thought about when designing the classes and interfaces, especially the Listener interfaces, was keeping it easy for a user to code their own listener under the two sub-listener interfaces. We had a discussion about if there was any other way to implement it that'll still be easy for a user-programmer to code for, and decided making the a pair of interfaces was the best path.*