**Work Plan – 31st October**

The blue colour text shows the updates that were made to the work plan during the design process. A list of the functional and non-functional system requirements is present after the work plan for reference.

Deliverable 1:

We created the functional and non-functional system requirements, which took around an hour. We got together as a group and created the domain model in 30 minutes. The Use cases, requirements-level sequence diagrams and implementation-level sequence diagram each took one hour. We separated the work of the source code for the 3 different prototypes and it took a total of 5 hours.

The food truck management system was written in Java, Android, and PHP in order to meet non-functional requirement #9. The other non-functional requirement that was addressed in this deliverable was #8 related to the persistence of the system. The produced code only contained the use case implementation for ordering. Thus, the functional requirements of #3, #4, #5 and #6 were addressed. The functional requirements #1 and #2 relating to the use case of scheduling staff, were left to be completed in a later deliverable. It was also not of particular concern to address functional requirements #11 and #12 as this was only a prototype program.

All of this was done by October 17th.

Deliverable 2:

        The block diagram shows the overall architecture of our software. It shows the relationships and dependencies between the main systems. We only show the systems on a global level, without writing details about those so it took around 4 hours to create.

        The class diagram is a more detailed design of how the different classes interact with each other and the methods they use. We need the domain model and use cases to help create the class diagram. Having these in hand, this took around 2 hours. We created the class diagram for the desktop application.

        These were done by October 31th.

Deliverable 3:

Once we know what functions we want to have in place and what they should perform, we will create unit tests for all the simple functions we create, which should not take more than an hour. Some functions will use many simple functions inside of them to perform something bigger and more important in relation to our program. We will create component tests for those, which should also take around an hour. Finally, we will create tests on a global scale of the software to see if our main functionalities work and if we reach our goals regarding the program. The systems tests should be faster to create than the component tests, and even faster than unit tests because we are working on the most global scale and we have fewer things to test. These tests stack up to each other. If the unit tests are not successful, then the component and system tests will not be either.

Performance/stress testing will be done after all tests pass. We will try to run our program with large amounts of objects and see how well it performs. This test should take the least amount of time, so around 30 minutes.

This deliverable largely concerns itself with creating tests for the program. Testing relates to the non-functional requirement #12 while also providing physical evidence that all of the functional requirements #1-#6 have been properly met.

These should be done by November 14th. As this deliverable is due within the next two weeks, the group members are aware of their schedules. It has been decided that a meeting will be held on the weekend of November 6th-7th to begin discussing the deliverable and allocating tasks for each of the members. Each member has mostly focused on one platform implementation until this point and this production method will continue for the testing stage.

Deliverable 4:

        The release pipeline will require collaboration between all of our team members as it will integrate all the different tools of our software and release it to the user. It should show all the different stages of handing the program from the team to the user. It shows which stages trigger others automatically or manually. For example, once all the tests and builds are successful, the test stage will manually trigger the release stage. An expected time of 3-5 hours will be dedicated for this.

        The pipeline plan should be done by November 21st.

Deliverable 5:

        Once the pipeline plan is complete and we have all other documents and systems created for our software, we will be ready to start preparing our presentation. We need to have a clear understanding of our program in order to present it to others and we need to encapsulate every part of it. This is expected to take the longest amount time so we will dedicate 7-10 hours for it.

The presentation material should be done by November 27th so we have time to get prepared for the actual presentation.

Deliverable 6:

        After the presentation is complete, we will take as many critics into account to improve the last bits of our program before submitting the full implementation. We will use an estimate of 2-3 hours for this part.

The final source code should be done by December 15th.

Reference Functional System Requirements

The system shall:

#1 - High Priority: Keep track of staff (i.e., their schedules and roles).

#2 - High Priority: Include functionality to manage staff (add, modify, remove).

#3 - High Priority: Keep track of available equipment (grills, fryers, etc.)

#4 - High Priority: Keep track of available ingredients (meat, lettuce, cheese, etc.)

#5 - High Priority: Keep track of menu items and their popularity.

#6 - High Priority: Handle customer orders if there are enough ingredients for the selected menu items and automatically update remaining ingredients in the case of a successful order and increase the recorded popularity of items ordered.

Non-functional System Requirements

The system shall:

#7 - High Priority: Support laptop/desktop machines, mobile devices, and web browsers.

#8 - Medium Priority: Support persistence, with changes from previous sessions being stored locally.

#9 - Medium Priority: Perform operations swiftly in a manner which minimizes user frustration.

#10 - High Priority: Ensure that the system is trustworthy in such a way that no data can be lost in the case of software malfunction.