**Sprint Planning:** GotoGro-MRM

**Team Details**

|  |  |
| --- | --- |
| **Team Name:** | MSP 14 |
| **Tutorial:** | Tue 2:30 ATC325 |
| **Tutor:** | Dr Kaberi Naznin |

|  |  |
| --- | --- |
| **Members:** | |
| Dylan Jarvis | 102093138 |
| Rabya Tayal | 103144215 |
| Simon Tran | 103602807 |
| Thomas Babicka | 103059885 |
| Cody Cronin-Sporys | 103610020 |
| Nicholas Dyt | 101624265 |

**Selection Factors**

The backlog items refined from task 02P were selected for sprint 1 in order of importance to the project. This importance, however, can be qualified in numerous ways. To ensure each team member is on the same page, a list of agreed factors and their justifications is shown in **Table 1**.

**Table 1. Justification of Selection Factors**

|  |  |
| --- | --- |
| **Factor** | **Justification** |
| Feature Dependency | Feature dependency was agreed to be the most important factor, especially in the early stages of the project. Since the features developed in sprint 1 are likely to be the most fundamental, everything that comes later is likely to depend on them. This will be used as the primary deciding factor |
| Business Value | Business value is inherently important, as this is primarily defined by the client and will be what the client looks for when the project is delivered. That said, business value is harder to quantify than dependency leaving it as s secondary deciding factor. |
| Development Effort | Development effort is a minor consideration to be made when engaging in the sprints. Assuming feature dependency and business value are approximately equal, development effort could be used as a discriminator. In general, the task that takes less effort for equal value will be prioritised. |
| Risk | Risk is another minor factor which uses the same logic as development effort. All things equal, the item of lower risk will be prioritised. |

As noted, the progression of factors in terms of weighted importance is:

**Dependency -> Business Value -> Development Effort -> Risk**

Timeline was not an included factor. This was on purpose as the agile development method does not place as much importance on a stringent timeline as other methods. Furthermore, it was agreed that feature dependency overlaps with timeline, in that the dependencies loosely define the chronological order in which tasks must be done.

Another minor factor considered was expertise, with the justification being that tasks the group had more knowledge around would be prioritised. Discussion yielded that for the scope of the project, we had enough knowledge within the group to achieve the tasks via delegation rather than having to specifically compensate for gaps in expertise.

Though touched in the justification of each factor, the definition of high-low classification under each factor is summarised in **Table 2**.

**Table 2. Definitions of High-Low Affinity of Each Factor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Factor** | **High** | **Med** | **Low** |
| Feature Dependency | 3 or more features require this as a prerequisite | Only 1 or 2 features require this as a prerequisite | No other feature requires this feature as a prerequisite |
| Business Value | The client has directly specified this feature as a requirement | The feature has not been specifically requested, but it is required to better meet another direct requirement | The client has not specifically requested the feature |
| Development Effort | Development effort is minimal in hours, implementation knowledge is to a high level | Development effort is average but there is some research or trial and error required to implement | Development effort is high in hours including time required to research implementation |
| Risk | Feature is low risk: Easily replaced and-or reimplemented, not critical to the function of the program, any team member could implement it. | Feature is medium risk: Somewhat replaceable, is required for program function but cannot be easily damaged. Only 1 or 2 team members can implement it easily. | Feature is high risk: Irreplaceable and critical to program function. Only 1 team member knows how to implement it. |

Using the following factors and justifications a more accurate selection can be made for the most important backlog items.

**Product Backlog Selection**

According to the documentation in task 02P, it was estimated that the team would only be able to the database backend. This estimation changed with scope clarification as well as task breakdown and discussion, revealing that the team believes we have the capacity to tackle the entire backend as well as some of the basic UI interaction endpoints. **Table 3** summarises the highest priority backlog items that can be developed in sprint 1.

**Table 3. Product Backlog Items**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Feature Dependency** | **Business Value** | **Development Effort** | **Risk\*** |
| Sales Record Table | High | Highest | High | Med |
| Inventory Table | High | High | High | Med |
| Member Table | High | High | High | Med |
| Add sales record UI | Med | High | Med | Med |
| Add/mod new member UI | Med | Med | Med | Med |
| Add/mod new item UI | Med | Low | Med | High |

\* Note, “High” affinity for the risk factor means the feature has low risk associated with it.

**Work Breakdown Structure (WBS) Justification**

Taking the overarching items from the backlog and breaking them down into achievable tasks gave the team a very solid idea of exactly what must be done. The following tables summarise the breakdown of each item.

**Table 4. Inventory Table Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 1 | Critical | Inventory Table | Create Item Table | Item table with data validation - NOT NULL, etc - which records item details for company inventory | - | Dylan | 0.3 |
| 2 | Major | Inventory Table | Create Queries to Add Item | Simple query to add item, will be triggered by visual fields on the UI | 1 | Dylan | 0.3 |
| 3 | Minor | Inventory Table | Create Queries to Delete Item | Simple query to delete item, will be triggered by visual fields on the UI | 1 | Dylan | 0.3 |
| 4 | Minor | Inventory Table | Create Queries to Modify Item | Simple query to modify item, will be triggered by visual fields on the UI | 1-2 | Dylan | 0.3 |

**Table 5. Member Table Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 5 | Critical | Member Table | Create Member Table | Member table with data validation - NOT NULL, etc - which records member details | - | Dylan | 0.3 |
| 6 | Critical | Member Table | Create Queries to Add Member | Simple query to add member, will be triggered by visual fields on the UI | 5 | Dylan | 0.3 |
| 7 | Minor | Member Table | Create Queries to Delete Member | Simple query to delete member, will be triggered by visual fields on the UI | 5 | Dylan | 0.3 |
| 8 | Minor | Member Table | Create Queries to Modify Member | Simple query to modify member, will be triggered by visual fields on the UI | 5-6 | Dylan | 0.3 |

**Table 6. Sales Record Table Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 10 | Critical | Sales Record Table | Create Sales Table | Sales table with references to both the items and members table | - | Dylan | 0.5 |
| 11 | Critical | Sales Record Table | Create Query to Add Sale | Simple Query to add sale, will be triggered by UI interface basically simulating a POS machine. When sale is added inventory of the item should decrease by the amount bought | 1, 10 | Dylan | 0.3 |

**Table 7. Add Sales Record UI Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 12 | Critical | Add Sales Record UI | Input Field to Add Member ID | Member ID must be added before any items such that each item can be associated with the correct member in the sales table | 10-11 | Cody | 4 |
| 13 | Major | Add Sales Record UI | Buttons to Add Different Items to Sale | Buttons simulate the effect of a barcode being scanned or similar, adding | 10-12 | Cody | 4 |
| 14 | Critical | Add Sales Record UI | Button to Confirm Sale | Pushes all sales records to the sales table, trigger the decrementing inventory for given items | 11-12 | Nic | 2 |

**Table 8. Add/Mod New Member UI Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 15 | Major | Add/Mod New Member UI | Input Fields for All Member Details | Text checking on input fields to minimise the chance of invalid data being entered | 5-6 | Rabya | 4 |
| 16 | Major | Add/Mod New Member UI | Confirm Button | Button to confirm the member details, checks the inputs then sends it to the member table | 15 | Nic | 2 |
| 17 | Critical | Add/Mod New Member UI | Autoincrementing Member ID | When the confirm button is pressed the member ID is automatically generated and added to the database | 16 | Rabya | 2 |
| 18 | Major | Add/Mod New Member UI | Viewport to View Members | Snapshot of the member table, needs to be able to be filtered by search interface | 5 | Rabya | 2 |
| 19 | Major | Add/Mod New Member UI | Search Input Field | By typing member ID in and confirming, the viewport will display the member searched for (or nothing if no results found) | 18 | Simon | 2 |
| 20 | Major | Add/Mod New Member UI | Modify a Member Record | Selecting the searched member result will populate the text fields with saved data. Writing over these with new information and confirming will save over the old record with the new information | 8, 15-19 | Simon | 4 |

**Table 9. Add/Mod New Item UI Item Breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Num** | **Level** | **Product** | **Item** | **Description** | **Prerequisites** | **Team Member** | **Est. Time** |
| 21 | Major | Add/Mod New Item UI | Input Field to Add Item | Text checking on input fields to minimise the chance of invalid data being entered | 1-2 | Dylan | 4 |
| 22 | Major | Add/Mod New Item UI | Confirm Button | Pushes item record to the item table | 22 | Thomas | 2 |
| 23 | Major | Add/Mod New Item UI | Viewport to View Item | Snapshot of the item table, needs to be able to be filtered by search interface | 1 | Thomas | 2 |
| 24 | Major | Add/Mod New Item UI | Search Input Field | By typing item ID in and confirming, the viewport will display the item searched for (or nothing if no results found) | 23 | Thomas | 2 |
| 25 | Major | Add/Mod New Item UI | Modify a Member Record | Selecting the searched member result will populate the text fields with saved data. Writing over these with new information and confirming will save over the old record with the new information | 4, 21-24 | Nic | 4 |

Time estimates were made using previous experiences as best the team could manage. In total, this sprint is estimated to take 43.2 hours out of a scheduled 48 hours. This leaves a little extra room for complications or simply underestimating the timeline.

Further to that, sprint 1 is packed much more densely than sprint 2 intentionally so that if something ends up unfinished then it can be resolved in the next sprint. This is much better than the alternative of unfinished work at the end of sprint 2 with no more time left to do anything about it.

**Member Comments**

**Table 10. Member Comments**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Dylan | The sprint planning phase was very successful, the team is confident that the goals set are achievable within the week. |
| Simon | I feel that we utilised our sprint planning phase effectively to delegate goals to members in a manner that plays to our strengths as a team. |
| Rabya |  |
| Cody |  |
| Thomas |  |
| Nic | The sprint planning phase broke down all tasks into smaller sub-tasks and effectively delegated the tasks per group member. The sprint is now fully planned out and will ensure that it will be more easily completable within the timeframe. |