NORHT-WEST UNIVERSITY

VAAL TRIANGLE CAMPUS

IT PLAN

ITRI 614

Oratile Seloro – 35834609

Sarah Masu – 32850123

Chriselda Mathebula – 34250964

Mduduzi Msiza – 26905647

Wilco Tromp – 34531033

Jevon Gounden – 37609866

Rorisang Serapelo – 35406895

Melissa Roodt – 33509735

Table of Contents

[1 Overview 1](#_Toc166580612)

[2 Document Organisation 2](#_Toc166580613)

[3 Goals and Scope 2](#_Toc166580614)

[3.1 Goals 2](#_Toc166580615)

[3.2 Non-Goals 3](#_Toc166580616)

[4 Technical Process 3](#_Toc166580617)

[4.1 Methodology 3](#_Toc166580618)

[4.2 Tools and Techniques 7](#_Toc166580619)

[4.3 Internal Artefacts 7](#_Toc166580620)

[5 Deliverables 8](#_Toc166580621)

[5.1 Primary Deliverables 8](#_Toc166580622)

[5.2 Secondary Deliverables 9](#_Toc166580623)

[6 Risk Management 9](#_Toc166580624)

[6.1 Risk Tracking and Management 9](#_Toc166580625)

[6.1.1 Risk Attributes 10](#_Toc166580626)

[6.1.2 Risk Management Process 11](#_Toc166580627)

[7 Scheduling and Estimates 11](#_Toc166580628)

[7.1 Project Schedule 11](#_Toc166580629)

[7.2 Work Breakdown Structure 12](#_Toc166580630)

[7.3 Resource Allocation 14](#_Toc166580631)

[7.4 Estimates 14](#_Toc166580632)

[7.5 Tracking and Schedule Changes 15](#_Toc166580633)

[8 Measurements and Metrics 15](#_Toc166580634)

[9 Appendix 16](#_Toc166580635)

[9.1 Project Plan Doc Additional Information and Resources 16](#_Toc166580636)

[9.2 References 16](#_Toc166580637)

**Table of Figures**

[Figure 1: Agile Scrum Method 6](#_Toc166580638)

[Figure 2: Scheduling Gantt Chart 12](#_Toc166580639)

[Figure 3: Project WSB 13](#_Toc166580640)

[Figure 4: Project Tabular WSB 13](#_Toc166580641)

[Figure 5: GitHub Resource Allocation 14](#_Toc166580642)

[Figure 6: GitHub Resource Allocation 15](#_Toc166580643)

**Table of Tables**

[Table 1: Goals 2](#_Toc166580644)

[Table 2: Non-Goals 3](#_Toc166580645)

[Table 3: Tools and Techniques 7](#_Toc166580646)

# Overview

Climate change has a big impact on agriculture across the globe. The South African Government is in search of new and innovative methods for predicting, detecting, and preventing droughts. Our application will focus on using a sustainable AI-driven system that is comprehensive and can be integrated into agricultural use, with the main goal of enhancing drought resilience and resource management within South Africa. To do this, we will implement a web API that is therefore accessible from most devices with an internet connection.

The team has been working together with the customer and has established that as mentioned above, making use of a web API is the best course of action as it will cater to the majority of potential future customers. The system will work by collecting information from sensors within the fields of farmers as well as weather patterns which we will collect from open-source satellite information. By using machine learning algorithms, we will teach the AI how to read the weather patterns and add sensor data to give an accurate forecast about if there will be a drought, and thus also inform the farmer, stakeholders, and policymakers that there is a drought on its way, and they need to start preparing.

Since the system is web-based it is not dependent on specific software and will be usable from almost all mobile devices and computers with internet connection. The farmers are the most important end customers for this application as they are the ones that will be affected the earliest and the hardest should there be a drought. It is therefore important that we receive accurate and consistent feedback from the farmers, but feedback from stakeholders and policymakers is also important. The feedback we receive from them is what we will use to improve the learning of the AI and the system in general.

The entirety of the project in the agreed scope and goals will be completed within the given time frame of April until October. The team will schedule features and milestones appropriately with signoff from the project sponsor to complete the project during the time frame and will need to account appropriately for breaks, days off, and risks such as team members becoming sick, all of which affect the schedule. The development of the system is responsibly shared equally among the team members, according to their strengths and weaknesses.

# Document Organisation

This document is meant to provide an overview of the drought prediction and detection system. It will cover the goals and scope of the project, the technical process, deliverables, risk management, scheduling and estimates, and measurements and metrics.

Certain sections of this document use Priority values to convey the importance of certain features/goals. The priority values are on a scale of P1 to P3. An explanation of the priority values follows:

* P1 is the topmost priority and is necessary for a viable product. Without these items, the product should not be released.
* P2 items are important and impact the quality of the product, but the product would still be shippable without items in this category.
* P3 items are nice to have but can be dropped if necessary. They are not required for a quality shippable product. This category includes enhancements.

Other tables in some sections may use different measurements related to their specific purpose or topic (i.e. the risk scoring table contains impact and probability values). The values for those tables will be defined in a description directly above the table.

# Goals and Scope

Defined goals provide primary objectives for the project and help define the scope. The following two sections specify this project’s prioritised goals and a series of non-goals with explanations, to clarify the scope, direction, and intentions of the project.

## Goals

Table 1: Goals

|  |  |  |
| --- | --- | --- |
| **Goal** | **Goal Description** | **Priority** |
| 1 | Provides timely alerts to stakeholders about impending drought conditions to allow for initiative-taking measures. | P1 |
| 2 | Evaluate potential risks of drought in various sectors, such as agriculture and water resources. | P2 |
| 3 | Assists farmers in making informed decisions concerning crop selection, irrigation practices, and land management. | P1 |
| 4 | Facilitates water management strategies to mitigate drought effects on water supplies. | P2 |
| 5 | Improves forecasting capabilities. | P1 |
| 6 | Implements strategies that adapt to changing climatic conditions. | P2 |
| 7 | Ensures that vulnerable populations receive adequate support and assistance during droughts. | P3 |

## Non-Goals

Table 2: Non-Goals

|  |  |  |
| --- | --- | --- |
| **#** | **Non-Goal** | **Reasoning** |
| 1 | Weather forecasting | While the system may use weather data as inputs, its primary focus is on long-term trends and impacts rather than short-term weather predictions. |
| 2 | Climate change mitigation | Since the system contributes to understanding the impact climate change has on drought frequency and intensity, its primary goal is to monitor and predict drought conditions rather than directly mitigate climate change. |

# Technical Process

## Methodology

In the selection process for a software methodology, our team considered a variety of options. The following software methodologies were considered:

* Waterfall Model: This is a linear and sequential model, implying that the next development phase cannot occur until the previous phase is completed and upon completing a phase one cannot simply iterate back. This methodology is easy to use and understand, however, it will not be used due to the following reasons:
  + No overlapping phases, implying each phase must be completed before the next iteration.
  + Testing comes late and difficult to make changes.
* V-Model: The Verification and validation model executes processes sequentially, and software development and testing take place at the same time. The advantages are that defects are detected early on; however, this methodology will not be used due to the following reasons:
  + Not suitable for complex, ongoing projects and projects where there is uncertainty in the requirements.
* Incremental Model: The model divides software development processes into several increments and the same phases are followed each increment. The advantages are that the detection of errors is easy to identify, however, this methodology will not be used due to the following:
  + Complete requirements of software must be available from the start.
* RAD Model: Rapid Application model is based on prototyping and iterative model with less or no planning. Reduced development time, flexibility, suitable for small projects is a great advantage however this methodology is not suitable due to the:
  + Difficult to manage, and not suitable for complex projects with large life spans.
* Iterative Model: Software is developed by following several iterations, each iteration implies following the development process, this is repeated until the project is completed. Iterations allow for quick error and bug handling; however, this methodology is not suitable due to the following:
  + Not suitable for small projects, difficult to estimate the end date for the project, constantly changing requirements implies frequent change to software.
* Spiral Model: Each “spiral” loop is a phase in the development process. Each phase contains determining objectives and alternative solutions, identifying, and resolving risk, developing, and testing, and reviewing and planning the next phase. The advantages are that it is suitable for large and complex projects, and continuous customer feedback implies greater satisfaction. However, the model is not chosen due to the following:
  + Requires more documentation than others, success of the model depends highly on the risk analysis phase, a high-cost model.
* Prototype Model: Protypes of projects are created initially and developed upon. The advantages are that it is good when the general objective of creating software is known but not the input, processing, or output. However, the methodology is not suitable due to:
  + Compromises can be seen in the prototype version; end-users might want to install small fixes on the prototype instead of a rebuilt version.
* Agile Model: Follows an iterative development method, whereby in each iteration a small and easily manageable task can be completed within a couple of weeks. This focuses on customer satisfaction and making quick changes during the middle of the development process. It has different models: scrum, crystal, feature-driven, lean and XP models. The advantages are dual programming which decreases errors, projects are completed in a short time frame, flexible to developers, realistic and managed easily. The disadvantages are a lack of documentation and dependence on customer representative information to build software correctly.

Considering the above advantages and disadvantages of the listed methodologies, the team reached a consensus on using Agile methodology. The model chosen was Scrum. Based on Peek (2023) the scrum model breaks the project down into sizable chunks called sprints. Agile scrum methodology is an industry norm methodology that produces projects quickly. This methodology relies on incremental development with each iteration commonly being within two to four weeks. The focus is to build the most important features first and deliver a potential product. More features are added in subsequent sprints and are adjusted based on stakeholder and customer feedback. The agile scrum has five phases: initiation, planning and estimation, implementation, reviewing and releasing. Figure 1 displays the phases of agile scrum.

Figure 1: Agile Scrum Method

A diagram of a software project

Description automatically generated

The benefit of agile scrum is it focuses on fast production since each goal must be completed within each sprint. It also requires frequent planning and goal setting, which increases team productivity on focus, flexibility, and adaptability where each sprint customer feedback is received and adapted in the next iteration. (Peek, 2023). The disadvantage is that it requires an expert in the company to ensure the principles are applied correctly. Agile scrum methodology involves precise execution and could result in problems if not done properly (Peek, 2023). The team has undergone agile training and is more than equipped to develop the project using this methodology.

The team will implement the Agile methodology in a two-weeklong iteration, which will provide for approximately eight iterations per semester. Given that each iteration has 5 phases, the team has agreed to a two-weeklong iteration which should provide enough time to go through each phase and receive frequent client feedback to produce a valuable project. The schedule will be evident and documented in the project schedule, which will be managed in the team spreadsheet.

## Tools and Techniques

Table 3: Tools and Techniques

|  |  |
| --- | --- |
| Project Management Requirements | Tools and Techniques |
| Version control | **Git & GitHub** – The team has experience with using git & GitHub, GitHub allows for easy version control of code in a large team in a private repo. |
| Project information website | **GitHub Pages** – Allows us to freely host a project website on GitHub using HTML and CSS. |
| Scheduling software | Sheets – The team is already using Sheets as a scheduling method. The use of Sheets allows for easy and fast scheduling and the use of equations can help with related tasks such as time tracking. |
| Team communication | **Teams** – The team is already using Teams as a communication method. Teams provide the following advantages: no restriction on meeting time, sharing of documents and resources, scheduling of team meetings, and recording of meetings. |
| Task Management | **GitHub Kanban** – The team is experienced with using GitHub Kanban as task management software. GitHub Kanban is easy to use and can be set to meet the needs of the team, each member can be assigned a task and can update their tasks as they progress. |
| Documentation Storage/Management | **Google Drive (App) & GitHub (code) -**  Google Drive is convenient and free to use, it promotes easy storage of applications, and documentation, and tracks history to aid in project documentation.  GitHub is an industry norm for storing the code of the project in a safe and secure environment and promotes version control and branching which will allow the team to work coherently without breaking each other’s code. |
| Continuous Integration | **Jenkins** - is an open-source automation server. It aids in software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. |

## Internal Artefacts

Agile scrum artefacts are information that a scrum team and stakeholders use to detail the product being developed, the actions to produce it, and the actions performed during the project (Harris, 2024). The main artefacts are product backlog, sprint backlog and increments (Harris, 2024), they are described in detail below:

* Product backlog: a list of new features, enhancements, bug fixes, tasks, or work requirements needed to build a product. It’s compiled from input sources like customer support, competitor analysis, market demands, and general business analysis (Harris, 2024). This will be kept up to date as new information is available.
* Sprint backlog: a set of product backlog tasks that have been promoted to be developed during the next product increment. Sprint backlogs are created by the development teams to plan deliverables for future increments and detail the work required to create the increment (Harris, 2024). A Sprint backlog will be created by selecting tasks from the product backlog and breaking them into smaller actionable sprint items.
* Increments: the customer deliverables that were produced by completing product backlog tasks during a sprint. It includes the increments of all previous sprints. There is always one increment for each sprint and an increment is decided during the scrum planning phase. An increment happens whether the team decides to release to the customer. Product increments are incredibly useful and complementary to CI/CD in version tracking and, if needed, version rollback (Harris, 2024). Increments will be aligned to each backlog item, by creating a branch and build for each backlog item.

# Deliverables

Project deliverables are derived from the agreed-upon client artefact, along with process (agile) and project-specific artefacts. These are defined in sections 3, and 7 respectively.

## Primary Deliverables

* **Project Plan** - Document containing goals, technical process, deliverables, risk management, scheduling specification, and measurement metrics for the project.
* **Hosted Front-end Drought Monitoring Website** - Front-facing website that system users interact with to detect and monitor droughts.
* **Hosted Backend API and Middleware** - Hosted codebase that supplied endpoints that are consumed in the front end.
* **Domain model** - Document describing the system’s domain layout
* **GitHub Project Tracker** - GitHub Kanban board tracking and accounting for the division of human resources across the whole project.
* **Prototype and Final Project Presentation** – Presentation where the initial prototype, as well as the final product (at a later date), will be presented to the product owner for feedback and submission.
* **Technical Report** - Comprehensive documentation for the mechanistic view of the entire system.
* **Team Self-Assessments** - Self-written reports to track progress and project-involvement.
* **Project Reflection** - Reports are written when the project is complete to facilitate growth and lessons learned.
* **GitHub Code Base** - The entire system’s code base that holds code for all programmed parts of it.

## Secondary Deliverables

* **Project Prototype** - MVPs submitted to the product owner during the development phase for constructive feedback to improve the final system deliverable.
* **Software Architecture Document** - Document illustrating the architecture of the system, how everything is built and how it all interconnects to the product owner to aid in later maintenance.

# Risk Management

## Risk Tracking and Management

Risks are managed in a separately maintained Google Sheets file on the project Google Drive, which can be found [here](https://docs.google.com/spreadsheets/d/1qdQMhWWx40y-qRoIiaWd3fm_OrWZZYwLyCoenbwfDvI/edit?usp=sharing). Each risk within the spreadsheet will be described by the following attributes:

### Risk Attributes

1. **Risk Name**: This refers to the name or identifier of each specific risk that we have identified for the project. For example, "Software Compatibility Issues" could be a risk name.
2. **Probability**: This is the likelihood of each risk occurring, expressed as a percentage (from 0% to 100%). For example, if there's a 30% chance of encountering a particular risk, you'd enter "30%" in this column.
3. **Impact**: This column allows us to categorise the impact of each risk on the project from options like "low(value of 1)," "medium(value of 5)," or "high(value of 10)" to indicate the severity of the risk's potential consequences.

* *Low Impact*: Minimal and manageable consequences.
* *Medium Impact*: Potential to affect project performance but not halt progress entirely.
* *High Impact*: Significant consequences halting the progress of the project.

1. **Exposure**: This is a calculated value obtained by multiplying the probability of each risk by its impact value. It helps prioritise risks based on their potential impact on the project.

* Probability and Impact:

The probability of a risk refers to the likelihood of it occurring during the project. For example, a risk like "Data Accessibility and Quality" might have a 40% chance of occurring.

Impact, on the other hand, evaluates the severity of the consequences if the risk were to occur. For instance, this risk might have a high impact, meaning it could significantly disrupt the project's progress.

* Combined Effect:

Exposure is calculated by multiplying the probability and impact values together. So, for the "Data Accessibility and Quality" risk, with a 40% probability and high impact value (10), the exposure value would be 400%.

This high exposure value indicates that if this risk were to occur, its combined effect of probability and impact would be quite substantial.

1. **Classification**: Here, we categorise each risk according to its primary impact area. Options provided include requirements, costs, quality, and scheduling.
2. **Owner**: This column assigns responsibility for managing each risk to a specific individual in the project team.
3. **Mitigation Plan**: Here we outline the actions or strategies to prevent each risk from occurring in this column. This could involve proactive measures to reduce the likelihood of the risk.
4. **Management Plan**: This section details how we will handle each risk if it does occur. It includes steps to minimise the impact on the project and ensure it can proceed smoothly despite the risk.
5. **Status**: This column tracks the current status of each risk throughout the project lifecycle. We will use labels like "active," "inactive," "mitigated," or "managed" to indicate the current state of each risk.

### Risk Management Process

1. **Identification**: Risks will be identified through regular team discussions and stakeholder input.
2. **Assessment**: Each identified risk will be assessed based on probability, impact, and exposure, to determine its significance to the project.
3. **Prioritisation**: Risks will be prioritised based on their exposure value, ensuring that resources are allocated effectively to address the most critical risks first.
4. **Mitigation and Management**: Owners of identified risks will be responsible for implementing mitigation plans and managing risks throughout the project lifecycle.
5. **Monitoring and Review**: The risk management process will be continuously monitored and reviewed to ensure its effectiveness in addressing threats and changing project conditions.

# Scheduling and Estimates

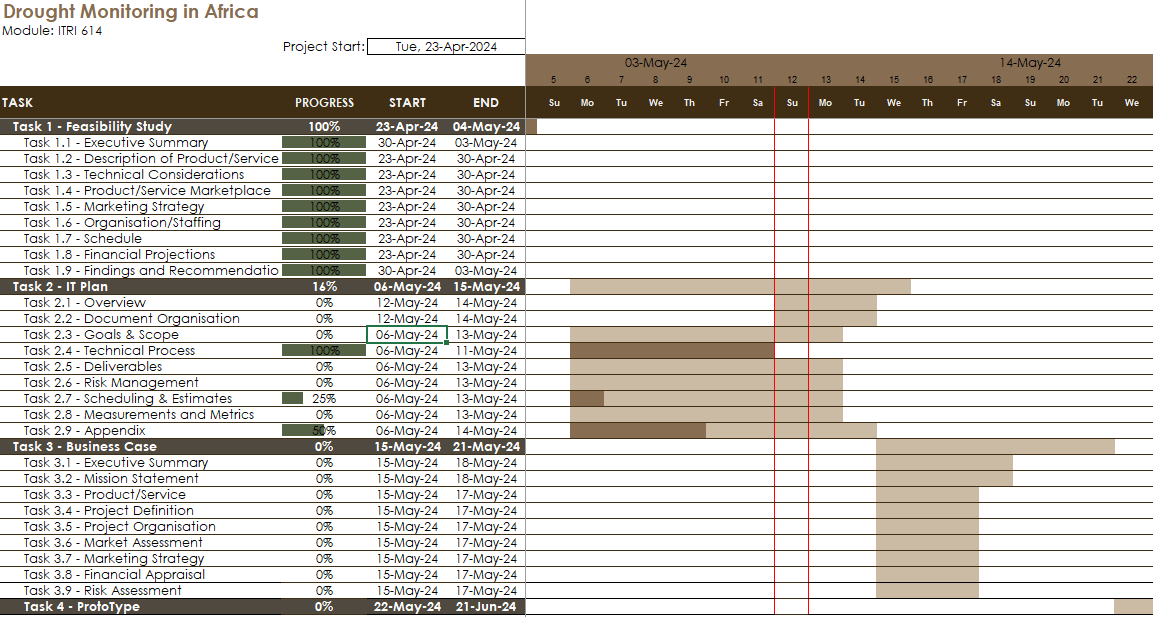
The schedule is managed using Google Spreadsheet as a scheduling tool, it can easily create and change the schedule using formulas, and the resources are managed using Word document. The schedule can be found on the team’s GitHub repository and or [here](https://docs.google.com/spreadsheets/d/1cEmcdVRQQ1Wq42YmGJ1cKYrTN_nRlugcogwNScpPr0Y/edit?usp=sharing).

## Project Schedule

The project schedule is set within the NWU semester period, this includes recess. The schedule needs to be able to adapt to unexpected changes. The schedule is set around the Agile Methodology, allowing the schedule to be easily changed. See a sample of the schedule below.

Sheets are used as a schedule management tool. The screenshot below is a sample of the screenshot. It includes all the tasks and iterations.

Figure 2: Scheduling Gantt Chart

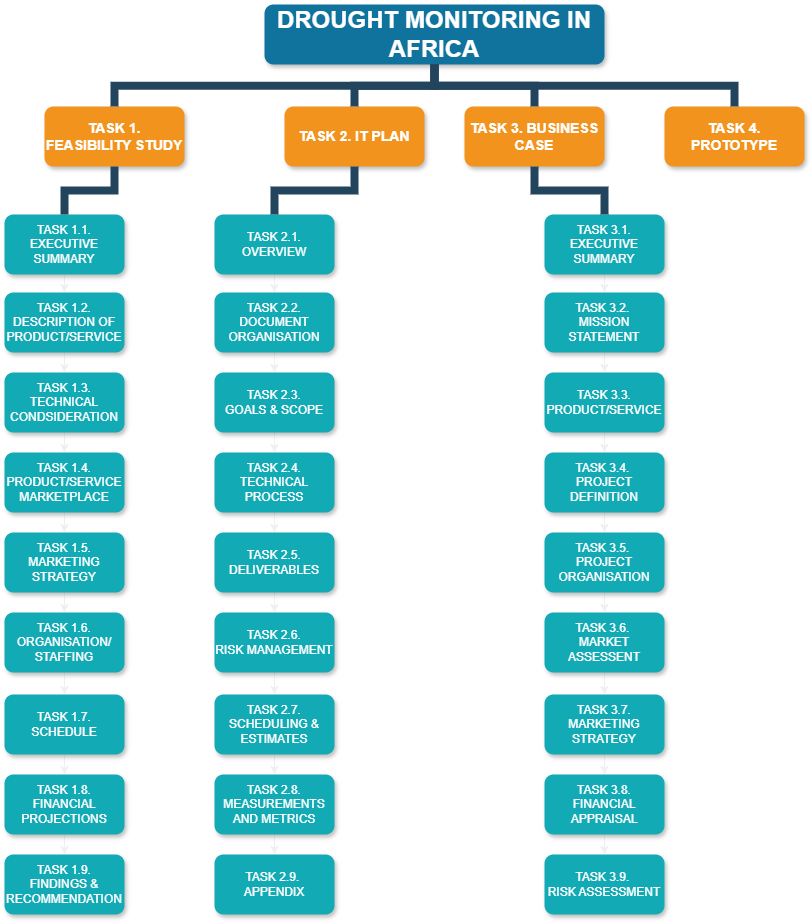


## Work Breakdown Structure

Work breakdown structure (WBS) is when large tasks are divided into smaller manageable tasks. The technique, commonly used in productivity techniques, displays tasks in a hierarchical view of high-level tasks, with the sub-tasks, in the projects.

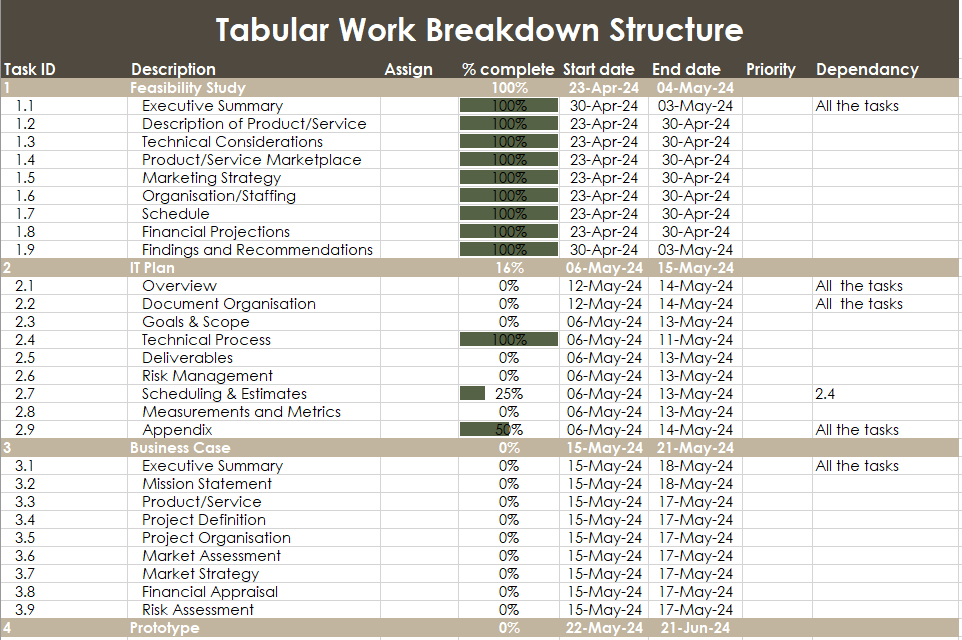
The image below is the WBS of the project, it will change over time as tasks are added or removed. The WBS is visually depicted using Draw.io, and the tabular structure is in the same Excel file as the Gantt file.

Figure 3: Project WSB



The tabular WBS uses the data from the Gantt chart and the data will synchronize when progress is made to the Gantt chart. If a task is updated, it will be reflected in the WBS table. If any task is updated it will need to be manually changed.

Figure 4: Project Tabular WSB



## Resource Allocation

All the resources for this project will be managed in GitHub.

The resources available are the personal resources that each team member possesses. These team members include Oratile Seloro, Sarah Masu, Chriselda Mathebula, Mduduzi Msiza, Wilco Tromp, Jevon Gounden, Rorisang Serapelo and Melissa Roodt.

Currently, the hardware resources consist of the team member’s personal laptops/desktops and mobile devices for development and testing.

The available software resources are Python, MATLAB, and existing South African drought data.

To allocate the resources, GitHub is used. An example of resource allocation is below in Figure 5. The link to the GitHub repository is [here](https://github.com/rorisang123/ITRI614-2024).

Figure 5: GitHub Resource Allocation

A screenshot of a computer

Description automatically generated

## Estimates

The estimations will be done on GitHub and in the Scheduling Sheets document. The estimations incorporate the time taken to complete a task, the tasks, and the sprints of the project. An example of the estimations is shown in Figure 6.

Figure 6: GitHub Resource Allocation

A screenshot of a computer

Description automatically generated

## Tracking and Schedule Changes

All of the tracking and schedule changes will be controlled by GitHub and Google Spreadsheets.

# Measurements and Metrics

# Appendix

## Project Plan Doc Additional Information and Resources

## References

GeeksforGeeks. 2023. Top 8 Software Development Life Cycle (SDLC) Models Used in Industry. <https://www.geeksforgeeks.org/top-8-software-development-models-used-in-industry/> Date of access: 8 May. 2024.

Peek, S. 2023. What is Agile Scrum Methodology? <https://www.businessnewsdaily.com/4987-what-is-agile-scrum-methodology.html> Date of access: 11 May. 2024.

Harris, C. 2024. Agile scrum artefacts. <https://www.atlassian.com/agile/scrum/artifacts#:~:text=The%20main%20agile%20scrum%20artifacts,archaeological%20ruins%20and%20ancient%20relics>. Date of access: 11 May. 2024.

Duke, R., n.d. *What is a Workdown Structure?.* [Online]   
Available at: https://www.workbreakdownstructure.com/  
[Accessed 11 May 2024].

Kukhnavets, P., 2024. *How to Create a Work Breakdown Structure (WBS) in Excel: a Detailed Guide [Alternative Included].* [Online]   
Available at: https://blog.ganttpro.com/en/how-to-create-wbs-in-excel/  
[Accessed 12 March 2024].