**Summary of Technical Details**

**Programming Languages:**

* **Python**: Used as the primary programming language for implementing models, managing data pipelines, and handling cloud-based operations.
* **JavaScript**: Used to create an access point to the OpenWeather data API, writing daily weather information (temperature, humidity, rainfall, wind speeds, and snow) to Google Sheets.

**Models Used:**

* **Transformer Models**:
  + Llama 3.2 Instruct
  + Sentence Transformer MPNet
* **Epidemiological Model**: Cropland connectivity
* **Markov Model**: Augmented with Black-Scholes configuration

**Databases and Storage:**

* **Google Sheets**: Stores weather data fetched from the OpenWeather API.
* **Hugging Face**: Cloud storage for large transformer models and version control during training.
* **Google Drive**: Storage of the trained and quantized transformers models for ease of access to the non technical users.

**Modules in Source Code (Located in src Folder):**

1. **Data Loader [dat\_load.py**:

Module to load and preprocess all the data used in the model apart from the custom data used to train the transformer models. This model loads the weather data from the google sheet where its being stored daily from the weather API. The module also ensures that the entire model is able to work or run in offline mode by temporarily storing weather data locally and retrieving the stored data when its unable to access the internet. That makes the entire ensemble model to run offline with data updated to the last day the user was online.

1. **Disease Epidemiology and Network [dis\_epi\_net.py]**:

This module contains the mathematical workings of the epidemiology model. A matrix network {Reference Project Manuscript for detailed explanation} The module give the disease index possibilities for every location covered by the epidemiology model giving the risk index of each and every disease possibility of occuring in an area

1. **Prompt Keyword [prompt\_keyword.py]**:

This module uses the concept of ngram to look for location keywords from users input prmpt and feeds the keyword location into the epidemiology model for computation to give the disease possibilities under different climatic conditions.

1. **TransformerRAGs [transformerRAGs.py]**:

This module intergrates the epidemiology information as a RAGs to the text generation transformer trained on numerous amount of potato variety data including their genetic traits and agromomic traits for tens of thousands of potato varieties. Its aso trained on genetic information. Potato diseases and agromonic management, social economic impacts of potato.

1. **Sentence-Transformer MPNet [sentencetransformerMPNet.py]**:

This module contains the code for scoring the outputs of the model against Sustainable Development Goals SDGs and to give a semantic rtelationship to project how interventions might be contributing towards SDGs on climate action, end hunger, biodiversity and other SDG targets. The scores are later use din the Markov model

1. **Continuous Chain Markov Model (Augmented with Black-Scholes)**:

The module is used to predict future states from the confidence scores of the in order to know if there are any breeding gaps that plant breeders can intervene.

1. **\_\_init\_\_.py**: Serves as an interface for integrating all models and importing them into the main.py file.
2. **Main.py** : Used to import from the \_\_init\_\_.py interface file and run the model

**Additional Files:**

* **cropland\_Script.js**: JavaScript file for fetching weather data from OpenWeather and writing it to Google Sheets.
* **llama-finetune-potato.ipynb**: Kaggle notebook for training the Llama 3.2 Instruct model.
* **merging-potato.ipynb**: Kaggle notebook for saving the transformer model while preserving training weights.
* **scoring-model-mpnet.ipynb**: Kaggle notebook for training the Sentence Transformer MPNet model.
* **potato\_WizardEFPR.py**: Main code file containing various models, including the epidemiology model and RAGs integration.

**Usage Instructions:**

1. Obtain the transformer models from cloud storage (Google Drive) using the link provided in the README.md file.
   * Download the GGUF file for ease of use (smaller size).
   * Unzip the models into the home directory of the ensemble model folder.
2. Follow the README.md file for installation instructions.
3. Run the following commands in the home directory:
   * python main.py
   * Alternatively, use python potato\_WizardEFPR.py to start using the model, this alternative iseasier and runs faster.