****

**PES University**

**Department of Computer Science and Engineering**

**UE21CS342BA1:Generic Programming**

**GP Project**

**Name: Sathwik HJ**

**SRN: PES1UG21CS544**

**Section: I**

**Name: Hemanth S Reddy**

**SRN: PES1UG21CS544**

**Section: D**

**Project Description:**

**File Type Detection:**

* It can identify various file types (e.g., JPEG, PNG, PDF, TXT, etc.) based on their magic numbers and file extensions.

**Additional Data Handling:**

* It provides functionality to handle additional data specific to certain file types. For example, it can count the number of lines and words in a text file, extract resolution from an image file, or read header information from a CSV file.

**Metadata Extraction:**

* It can extract metadata such as file size, creation time, last modified time, and file permissions.

**Binary File Support:**

* It includes support for extracting metadata from binary files, including the names, sizes, and creation times of individual files stored within.

**User Interaction:**

* It prompts the user to enter a file path, determines whether it's a binary file, and then performs the necessary operations to identify the file type and extract metadata.

**List of GP features:**

* **Abstract Base Class:**
  + FileTypeBase serves as an abstract base class for different file types, providing common interface methods.
* **Inheritance:**
  + Concrete file type classes (JPEGFileType, PNGFileType, etc.) inherit from FileTypeBase to implement specific file type functionalities.
* **Template Specialization:**
  + Template specialization is used for AdditionalDataHandler classes to provide specific implementations for handling additional data for certain file types.
* **Template Classes:**
  + FileTypeRegistry and BinaryFileTypeRegistry are template classes that manage file type definitions and metadata extraction.
* **Type Traits:**
  + std::is\_base\_of and std::is\_same from <type\_traits> are used to check type relationships and type equality.
* **Concepts:**
  + Concept DerivedFromFileTypeBase is defined to constrain template parameters to types derived from FileTypeBase.
  + Concepts are also used in template functions like isSpecificFileType.
* **Smart Pointers:**
  + std::shared\_ptr is used to manage ownership of file type instances.
* **File System Library:**
  + <filesystem> library is used for working with file paths and directory structures.
* **Stream I/O:**
  + <iostream> and <fstream> are used for file I/O operations.
* **Dynamic Polymorphism:**
  + Virtual functions in FileTypeBase allow for dynamic dispatch of methods in derived classes.
* **Error Handling:**
  + Error handling is implemented using standard error streams (std::cerr) and exceptions.
* **File System Queries:**
  + <sys/stat.h> is used for querying file system information such as file size and timestamps.

**Individual contributions:**

***Hemanth:***

***Sathwik:***

**Link to the Git repository:** [*Press Here*](https://github.com/HemanthsReddy/FileMetaDataAnalyzer.git) *(*[*https://github.com/HemanthsReddy/FileMetaDataAnalyzer.git*](https://github.com/HemanthsReddy/FileMetaDataAnalyzer.git)*)*

**List of known bugs**