**Patch Analyzer for ALLOT**

**Overview:**

The Patch Analyzer is a specialized tool developed to analyze unified diff/patch files for various parameters such as formatting, consistency, and adherence to coding standards. It provides a detailed analysis report and can send this report via email. The tool is integrated into a Flask web application, allowing easy file upload and analysis..

**Features:**

* **File Upload**: Users can upload unified diff/patch files for detailed analysis.
* **Analysis**: The tool thoroughly examines the uploaded files for formatting, consistency, and adherence to coding standards.
* **Email Notification**: Upon completion of the analysis, the tool dispatches an email notification with the comprehensive analysis report to a designated recipient email address.
* **Web Interface**: The Flask web application offers an intuitive interface for uploading files and perusing analysis results.

**How to Use the Analyzer:**

* **Uploading Files**: Visit the web interface, click on the "Choose File" button, select a unified diff/patch file, input a recipient email address, and click the "Submit" button to upload the file.
  + Only files with the ".diff" or ".patch" extension are accepted.
* **Analysis**: The tool will perform an in-depth analysis of the uploaded file, examining its formatting, consistency, and adherence to coding standards.
* **Email Notification**: Once the analysis is complete, the tool will send an email notification to the specified recipient email address, including the analysis report as an attachment.
* **Accessing Analysis Results**: The web interface will exhibit a success message indicating that the file was successfully uploaded Analyzed and Email sent. Users can also peruse the analysis report in the email dispatched to the recipient email address.

**Detailed Analysis by Patch Analyzer.py:**

**The Patch Analyzer tool offers the following checks only for .cpp and .h Coding Standards**:

* **Unified Diff Parsing:** The Patch Analyzer parses unified diff files to extract information about changes between different versions of source code files. It identifies the affected files and the specific changes made within those files.
* **Hunk Analysis:** For each file affected by a patch, the Patch Analyzer analyzes individual "hunks" within the diff to determine whether lines were added, deleted, or modified. It identifies the start and end points of each hunk to isolate the changes.
* **Line Change Detection:** Within each hunk, the Patch Analyzer detects the type of change made to each line (e.g., added, deleted, or modified). This information is used to understand the nature of the code modifications.
* **Code Style Checks:** The Patch Analyzer performs basic code style checks on modified lines, such as indentation and whitespace consistency. It ensures that the modified code conforms to the established coding standards.
* **Syntax Error Detection:** While not a comprehensive compiler, the Patch Analyzer may perform basic syntax checks on modified lines to detect obvious errors that could cause compilation issues.
* **Context Analysis:** The Patch Analyzer considers the context of code changes by examining the surrounding lines in each hunk. This helps in understanding the purpose and impact of the changes within the broader codebase.
* **Variable and Function Naming Conventions:** Although not exhaustive, the Patch Analyzer may check for adherence to variable and function naming conventions within modified lines to maintain code consistency.
* **Logging and Reporting:** The Patch Analyzer logs the results of its analysis, including details about each patch, the files affected, and the nature of the changes. It generates a report summarizing the analysis results for easy review.
* **Error Handling:** The Patch Analyzer includes robust error handling mechanisms to gracefully handle unexpected issues, ensuring that the analysis process is reliable and informative.
* **Performance Optimization:** The Patch Analyzer is designed to efficiently process large, unified diff files, optimizing resource usage and analysis speed to provide timely results.
* **Extensibility:** The Patch Analyzer is designed to be extensible, allowing for the addition of new checks and analysis features as needed for specific projects or coding standards.

Overall, the Patch Analyzer provides a comprehensive analysis of unified diff files, helping developers understand and review code changes efficiently and effectively.

**Limitations:**

* **Single File Upload:** Only one file can be uploaded for analysis at a time.
* Analysis Time: The analysis process, including sending out the email with logs, typically takes under a minute for a large file, and even less for smaller files.
* **File Size Limit:** There is a limit to the size of the file that can be uploaded for analysis - **Max Limit is 5MB**.
* **File Extension Requirement:** The uploaded must have the “.diff" or ".patch" extension.
* **Fixed Analysis Settings:** Currently, users cannot change the analysis settings. Future updates may allow customization based on project requirements.
* **No Result Display on GUI:** There is no provision to view the analysis results directly on the GUI. This may be considered for future enhancements.
* **Automatic Deletion of Uploaded Files:** Uploaded files and logs are automatically deleted from the server once the email containing the logs is sent to the recipient email mentioned in the GUI.
* **Update Policy:** The tool will be updated based on issues raised by end users and to incorporate new features or improvements.

**How to Access Application:**

The application's graphical user interface (GUI) can be accessed by opening a web browser, preferably Chrome, and entering the following URL in the Web browser:

[**http://192.168.0.80**](http://192.168.0.80)

**Feedback and Support:**

If you encounter any issues, have suggestions, or require assistance, please reach out to me at [**manu.m@thinkpalm.com**](mailto:manu.m@thinkpalm.com) or via Teams/Skype.