**Automated File Classifier and Safety Assessor for Digital Forensics**

**System Requirements**

The system makes use of several Python libraries for executing various functionalities:

1. **Flask**: A lightweight web development framework, Flask is used to build the web interface that allows users to interact with the system.
2. **os**: This built-in Python module is used for various operating system-dependent functionalities, particularly managing file and directory operations.
3. **shutil**: This Python library offers a range of high-level operations on files and collections of files, assisting in automated file management.
4. **werkzeug**: Used for ensuring security in file handling by providing secure file names.

**Design Decisions**

The choice of Python as the primary programming language was a deliberate decision based on its readability, extensibility, and vast array of standard and third-party libraries. This makes Python versatile for a wide range of tasks and particularly suitable for this project (Guttag, 2016). The agent-based architecture was another key design choice to ensure system modularity, which in turn enhances scalability and maintainability of the system (Gamma et al., 1994). The Flask framework was chosen to handle file uploads for its simplicity and robust security features (Grinberg, 2018).

**Approaches**

The development of the system embraced an iterative approach, refining and extending functionality in successive iterations. This strategy is in line with Agile methodologies, which prioritize incremental and flexible development (Beck et al., 2001). Classification of files, initially based on file extensions, could be enhanced in the future with sophisticated methods such as content analysis or machine learning models.

**Rationale**

The rationale for our design choices centered on creating a user-friendly, secure, and efficient system. Python’s simplicity and extensive library support aligned with these objectives (Guttag, 2016). The modular design allows for system scalability and easier maintenance, consistent with best practices in software design (Gamma et al., 1994). The Flask framework provides an easy and secure way to handle file uploads, contributing to user-friendliness and system security (Grinberg, 2018).

**Challenges and Approaches**

A significant challenge in the development process is ensuring the accurate classification of files. The current approach uses file extensions, which can be misleading if a malicious file is disguised with a benign extension (Casey, 2011). Further, ensuring secure file upload to prevent directory traversal attacks is a critical concern (Stuttard & Pinto, 2011).

**Graphical Designs**

In terms of visual representation, the system's design will include a class diagram illustrating the structure of the agents, and a sequence diagram to depict the interaction between the agents and the user.

**Explanation of the Agents**

1. **File Upload Agent**: This agent is responsible for accepting file uploads from the user through the web interface. It leverages Flask's capabilities to handle file uploads securely.
2. **File Classification Agent**: This agent's function is to classify the uploaded files based on their extensions. It categorizes files as either 'safe' or 'suspicious' based on a predetermined list of file extensions.
3. **File Movement Agent**: Post-classification, this agent moves files to their designated folders ('Safe' or 'Suspicious'), and within these, to specific sub-folders based on file type. It uses the Python 'os' and 'shutil' libraries for these operations.

**References**

1. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. Agile Alliance.
2. Casey, E. (2011). Digital Evidence and Computer Crime. Academic Press. ISBN 978-0-12-374268-1.
3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2.
4. Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media. ISBN 978-1-4919-6147-4.
5. Guttag, J. (2016). Introduction to Computation and Programming Using Python. MIT Press. ISBN 978-0-262-52962-4.
6. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook. Wiley. ISBN 978-1-119-96327-1.

**Sample Python Code for Agents**

**1. File Upload Agent**

The agent utilizes Flask's request object to handle file uploads.

*from flask import Flask, request*

*from werkzeug.utils import secure\_filename*

*import os*

*app = Flask(\_\_name\_\_)*

*app.config['UPLOAD\_FOLDER'] = 'uploads/'*

*@app.route('/upload', methods=['POST'])*

*def upload\_file():*

*if 'file' not in request.files:*

*return 'No file part'*

*file = request.files['file']*

*if file:*

*filename = secure\_filename(file.filename)*

*file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))*

*return 'File uploaded successfully'*

*if \_\_name\_\_ == '\_\_main\_\_':*

*app.run(debug=True)*

**2. File Classification Agent**

This agent classifies the uploaded files into 'safe' or 'suspicious' based on their extensions.

*SUSPICIOUS\_EXTENSIONS = ['exe', 'dll', 'pif', 'tmp', 'swf']*

*def classify\_file(filename):*

*file\_extension = filename.rsplit('.', 1)[1].lower()*

*if file\_extension in SUSPICIOUS\_EXTENSIONS:*

*return 'suspicious'*

*else:*

*return 'safe'*

**3. File Movement Agent**

After classification, this agent moves files to their respective directories based on the file type.

*import shutil*

*def move\_file(file\_path, category):*

*new\_folder = os.path.join(category, file\_path.rsplit('.', 1)[1].lower())*

*os.makedirs(new\_folder, exist\_ok=True)*

*shutil.move(file\_path, new\_folder)*

These agents will work together to handle file uploads, classification, and relocation, as part of the "Automated File Classifier and Safety Assessor for Digital Forensics" system.

**References**

1. Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media. ISBN 978-1-4919-6147-4.

2. Python Software Foundation. (2021). The Python Standard Library. Retrieved from https://docs.python.org/3/library/index.html.

3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2.