Test Document Validator

Certainly! Based on your requirements, here is a descriptive breakdown of the project and the steps involved:

# Problem Description

You need to create a tool that enables testers to validate documents, such as templates or email content, against a reference document or template. The tool should provide the following functionalities:

- Upload or copy and paste the reference document/template: Testers should be able to upload a document or copy and paste its content into the application.

- Upload the document to be validated: Testers should be able to upload the document they want to validate against the reference document/template.

- Confirmation and analysis: Once the documents are uploaded and confirmed, the tool should analyse both documents and identify discrepancies or variations in the content.

- Generate a report: The tool should generate a report summarizing the analysis and discrepancies found between the reference document/template and the tested document.

- Attach the report to the test evidence document: The generated report should be downloadable and attachable as evidence for further reference and sharing.

# 2. Tools and Technologies:

To build this software, you would need the following tools and technologies:

- Front-end Development: HTML, CSS, JavaScript, and a framework like React or Angular for the user interface.

- Back-end Development: A server-side programming language such as Python, PHP, or Node.js to handle document processing and analysis.

- Document Processing: A library or tool to process and analyse the document content. For example, you can use Python libraries like `python-docx` or `pdfminer` for Word and PDF documents respectively.

- Report Generation: A library or tool to generate the report, such as `pdfkit` or `JasperReports`.

- File Storage: A mechanism to store and manage uploaded documents, such as a file system or a cloud storage service like AWS S3 or Google Cloud Storage.

- Authentication and Authorization: If you want to add user authentication and access control, you can use frameworks like Firebase Authentication or OAuth2.

# 3. Timeline and Estimation:

The timeline required to learn the technologies depends on your existing knowledge and learning pace. Here is a rough estimate:

- Front-end Development: If you are new to front-end development, it may take around 2-3 weeks to learn the basics of HTML, CSS, and JavaScript and get familiar with a framework.

- Back-end Development: The time required to learn a server-side language and its associated framework can vary. It may take several weeks to a few months to gain proficiency.

- Document Processing and Report Generation: The time required to learn and implement document processing and report generation depends on the complexity of the documents and the chosen libraries or tools. It may take a few weeks to learn the basics.

Overall, considering learning time and development, you may need several months to complete the project.

For an individual working on this project, the estimated number of weekends needed would depend on the complexity of the project and the time you can dedicate each weekend. You can estimate the number of weekends by breaking down the tasks into smaller milestones and estimating the time required for each milestone.

# 4. Project Plan:

Here is a general plan for an individual working on this project:

## Learning Phase:

- Spend time learning HTML, CSS, and JavaScript for front-end development.

- Choose a suitable front-end framework and learn its basics.

- Learn a back-end language and framework for server-side development.

- Understand the basics of document processing and analysis libraries or tools.

- Explore report generation libraries or tools.

## Development Phase:

- Build the front-end interface to upload documents and display the analysis report.

- Implement the back-end functionality to process and analyse the documents.

- Develop the logic to compare the uploaded document with the reference document/template and identify discrepancies.

- Integrate a report generation library or tool to generate the report.

- Implement file storage mechanisms to handle uploaded documents.

- Ensure proper authentication and authorization mechanisms if required.

## Testing and Deployment:

- Thoroughly test the application, including document validation, analysis, and report generation.

- Deploy the application to a hosting server or platform so that it can be accessed live.

- Monitor the system for any issues and ensure the smooth functioning of the document validation process.