Fraunhofer IESE

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

GenerAltor

Capstone 2024

Software Architecture Document

1. Introduction

1.1 Project Motivation

GenerAItor is an advanced AI-powered document analysis tool developed to automate the extraction, processing, and structuring of data from various document types, including PDFs, DOCX files, and images. Leveraging AI models such as ChatGPT, Mistral, and Claude.ai, as well as proprietary tools like "AutoExtractor," GenerAItor allows users to streamline document management processes with minimal manual intervention. It enables users to upload, process, and structure document data quickly and with minimal effort. The tool is designed

for use by business partners, service providers, and customers looking to automate data extraction workflows efficiently and securely. GenerAItor serves a range of stakeholders, including business partners who configure custom services and end-users who use these services to streamline document data extraction.

1.2 Overall Project Goals

The overall project goals center around enhancing the frontend and backend functionalities of GenerAItor to provide an intuitive, efficient, and user-friendly experience. On the frontend, the focus is on developing a modular, interactive tutorial system that automatically engages new users upon login. This system will utilize cookies to differentiate between new and returning users, offering a seamless onboarding experience for first-time users while allowing returning users to manually initiate the tutorial through a dedicated button. The tutorial will incorporate navigation features such as a 'Next' and ´Back´ buttons for progressing through fields with concise explanations and a 'Skip' option to bypass certain steps, maintaining continuity in subsequent stages. Additionally, a comprehensive help page will provide FAQs, step-by-step instructions, troubleshooting tips, and support contacts, ensuring that users have access to all necessary resources for effective service creation.

For the backend, the focus is on enhancing user input by **proposing better field names** and implementing **intelligent pre-selection mechanisms**. These improvements aim to:

1. **Meaningful Field Names**: Provide clear, intuitive labels for data fields to reduce ambiguity and cognitive load on users. Thoughtfully designed names ensure users immediately understand the purpose of each field, making interactions more seamless.
2. **Intelligent Field Pre-selection**: Highlight the most relevant fields based on user context or previous inputs. This reduces manual effort and minimizes errors, as users can focus on completing only the necessary actions.

These measures prioritize usability and efficiency, allowing users to navigate and utilize the system more confidently. By streamlining input processes and enhancing field clarity, the backend design fosters a smoother and more accurate data entry experience.

1.3 Project Strategy

The project strategy focuses on developing a scalable, user-friendly, and automation-focused solution for document management using advanced AI techniques. The core goals include enhancing the user experience by designing an intuitive and accessible user interface and streamlining workflows to improve efficiency. Automation is central to our strategy, as we leverage state-of-the-art LLMs to intelligently select and organize fields within documents, implement dynamic field preselection mechanisms, and develop context-aware tutorial systems that adapt to user interactions.

Our collaborative approach emphasizes stakeholder engagement at every stage of development. Regular iteration meetings with feedback loops are crucial for aligning our deliverables with the expectations of key users, including academic supervisors, and industry partners. We prioritize understanding their pain points, such as reducing manual intervention, ensuring data accuracy, and simplifying complex workflows. The iterative nature of the project ensures incremental improvement, starting with core backend functionalities for field pre-selection and proposing better field names. Subsequent phases focus on frontend enhancements, such as UI optimization and creating comprehensive help pages, ultimately culminating in system validation and the creation of a marketing video to showcase the project's value.

This project represents a balance between academic rigor and real-world application, aligning technical innovations with user needs. By dividing responsibilities among team members, maintaining open communication, and adhering to a structured project plan, we aim to create a solution that not only showcases the potential of AI in document management but also demonstrates our capability to tackle complex software engineering problems as a cohesive and collaborative team.

1.4 Document Scope and Goals

The Software Architecture Document (SAD) outlines the architectural framework for the GenerAItor system, focusing on its design principles, structural organization, and key functionalities. The document serves as a foundation for understanding the system's architecture and as a communication tool among stakeholders, including developers, supervisors, and end users. It ensures that the system's design aligns with its functional and non-functional requirements, promoting scalability, maintainability, and usability. Furthermore, the SAD provides a comprehensive reference for future development phases, system updates, and maintenance activities, ensuring consistency and coherence in the implementation. By articulating the system's architecture in detail, the document aims to bridge the gap between technical development and user needs, fostering alignment across all project phases.

1.5 References to detailed document

The detailed requirements outlined in this document are derived from a comprehensive requirement engineering process that involved extensive analysis and documentation. Key inputs were drawn from the literature and resources provided by the Fraunhofer team, which offered valuable insights and technical guidance. Additionally, the iterative discussions held during each meeting played a critical role in refining the requirements; inputs from the Insiders team were particularly instrumental in addressing practical challenges and aligning the system with real-world needs. Our team's research efforts, combined with collaborative brainstorming sessions, further enriched the requirements, ensuring that the system design is both innovative and grounded in practicality. This integrated approach to gathering and

validating requirements has ensured a robust foundation for the architecture and subsequent implementation.

1.6 Stakeholders

To ensure the success of the project, stakeholders were carefully selected based on their roles, expertise, and the value they could bring to the development process. These stakeholders were chosen to represent all critical aspects of the project, from technical guidance and academic oversight to end-user perspectives and client requirements. Their collaboration was vital in aligning the project's goals with practical applications and long-term sustainability. The key stakeholders are as follows:

Insiders Technologies: As the product owners, Insiders Technologies play a pivotal role in steering the project. They act as clients to the RPTU student team, providing decision-making authority, defining detailed requirements, validating deliverables, and facilitating negotiations to ensure the project remains on track and meets its objectives.

Fraunhofer IESE: Fraunhofer IESE serves as the mentor and supervisory body for the student team, offering guidance to ensure that the project adheres to high-quality standards. They provide essential resources, mediate communication between the student team and Insiders Technologies, and support smooth project progression by aligning theoretical insights with practical execution.

RPTU Students: The RPTU students are the core development team tasked with creating a fully functional product that meets all specified client requirements. Their responsibilities include conducting requirements analysis, designing and implementing features, managing project iterations, and coordinating team efforts to deliver results that align with stakeholder expectations.

Business Partners: Business partners represent end-users who interact directly with GenerAItor. Their role involves utilizing the system to develop tailored services that address their business needs and satisfy the demands of their customers, providing a real-world perspective on the system's functionality.

Customers/Service Users: Customers, or service users, are the ultimate beneficiaries of the automated document-processing services developed using GenerAItor. Their primary objective is to harness the system's capabilities for efficient and accurate document data extraction, ensuring improved workflow efficiency and reduced manual intervention.

This diverse group of stakeholders ensures a comprehensive approach to the project by combining technical expertise, academic guidance, and practical applications.

2. Architecture Drivers

2.1 Functional Requirements

The functional requirements for the GenerAItor system are divided into two key areas: UI/Frontend and Backend. These requirements ensure the system delivers a user-friendly experience while maintaining robust and efficient backend functionality.

2.1.1 UI/Frontend Functional Requirements

FUNC-FE-001: The system must include a tutorial feature designed to simplify the user onboarding process. This tutorial will provide step-by-step guidance to new users, offering relevant information for each phase of the service creation process to ensure tasks are performed efficiently and accurately.

FUNC-FE-002: A comprehensive help page must be developed to assist users in resolving common service creation issues. The help page should include FAQs addressing frequent problems encountered during the use of GenerAItor. It should also detail procedures for various tasks, ensuring users have access to clear, actionable solutions.

2.1.2 Backend Functional Requirements

FUNC-BE-001: The system must propose accurate field names for all extracted fields upon completion of the extraction process. This feature should enhance the usability and precision of the extracted data.

FUNC-BE-002: The system must provide a set of pre-selected fields unique to the type of document uploaded by the user. This preselection aims to minimize manual input and streamline the data extraction process.

FUNC-BE-003: The system must group extracted fields based on their relationships to improve data organization and usability. This grouping should create a logical structure, reducing cognitive load and facilitating efficient navigation of extracted information.

These functional requirements ensure the GenerAItor system delivers a seamless, intuitive experience on the frontend while maintaining accuracy, efficiency, and scalability on the backend.

2.2 Quality Requirements

The system's quality requirements are focused on ensuring usability, reliability, performance, and maintainability to meet the needs of all stakeholders effectively.

2.2.1 Usability

The system is designed to prioritize ease of use by providing a user-friendly interface and structured onboarding support. The onboarding process includes step-by-step tutoria

36ls tailored to specific features and workflows, allowing users to quickly become proficient in using the application. A detailed help page, including categorized FAQs, enables users to find answers to common queries efficiently.

2.2.2 Reliability

Reliability is ensured through consistent performance and robust error-handling mechanisms. The system is built to accurately extract and process data under various scenarios, maintaining stability even when dealing with diverse or complex document structures.

2.2.3 Performance

The system emphasizes high efficiency in operations, including quick data extraction, field preselection, and seamless tutorial activation. Backend processes are optimized to ensure responsiveness, even under heavy workloads or with complex document inputs, maintaining a smooth user experience.

2.2.4 Maintainability

The modular architecture of the system ensures that it can be easily updated, debugged, and expanded. This design supports future scalability while minimizing disruptions during upgrades or the addition of new features. Comprehensive documentation further aids in the efficient maintenance and evolution of the system.

2.2.5 Alignment with Stakeholder Expectations

Regular feedback loops with end-users and stakeholders are integral to the development process. These interactions validate that the system meets the expected standards of usability, reliability, performance, and maintainability, ensuring alignment with the needs of all stakeholders. Continuous collaboration fosters trust and ensures the system's relevance and effectiveness.

2.3 Constraints

2.3.1 Organizational Constraints:

Project Timeline:

The project must be completed by December 16, 2024, with several iteration meetings scheduled throughout October, November and December to monitor progress. Key dates include iterations meetings on October 1, October 8, October 22, November 5, November 19, and December 3, 2024. The final presentation on December 16, 2024, establishes a firm deadline, requiring careful planning and adherence to the timeline.

Team Formation and Collaboration

- **Team Composition**: The project team consists of 14 students, split equally into:
    - Front-End Team (7 members): Focuses on designing the user interface and ensuring an engaging user experience using tools like Intro.js.
    - Back-End Team (7 members): Handles the server-side logic, API development, and services creation using Kotlin and REST APIs.

- **Role Assignments**:
    - Each team member is assigned two distinct roles to maximize individual contribution and cover diverse skill areas.
    - Despite specific roles, the project is collaborative, with all members actively participating in multiple tasks, including brainstorming, implementation, testing, and integration.

This structure fosters shared ownership of the project, encouraging knowledge exchange and teamwork to ensure timely delivery of milestones and project goals.

2.3.2 Technical Constraints:

The project relies on the team's ability to work with the existing technology stack and infrastructure. However, limitations in available tools, frameworks, or integrations might restrict what can be implemented. To address these challenges, the team needs to carefully assess what is feasible and make efficient use of the resources and technologies at hand.

2.3.3 Processual Constraints:

As the team has limited experience, this may lead to challenges such as slower development, potential errors, and a longer learning curve. These factors could affect both the timeline and the quality of the final product. To mitigate these issues, the team should focus on collaboration, seek guidance when needed, and dedicate time to learning and improving throughout the project.

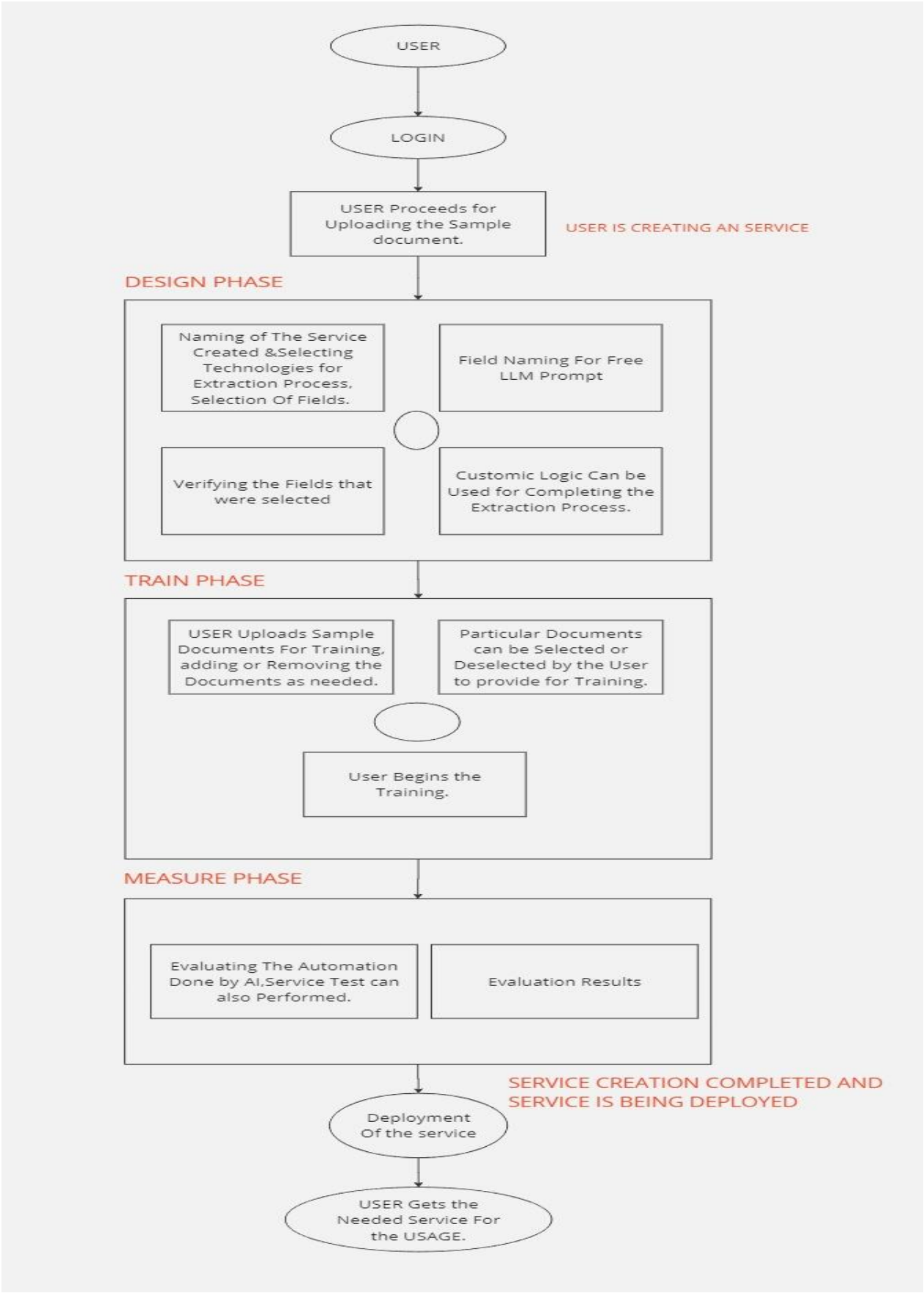
3. System Overview and Decomposition

3.1 System Structure

The system structure dictates the way various components of GenerAItor interact to ensure a seamless document processing workflow. It is composed of numerous layers, each of which is accountable for a specific function, including user interaction, service design, training, evaluation, and deployment. The seamless integration, accuracy, and reliability of real-world applications are guaranteed by this layered approach.

3.1.1 Existed System

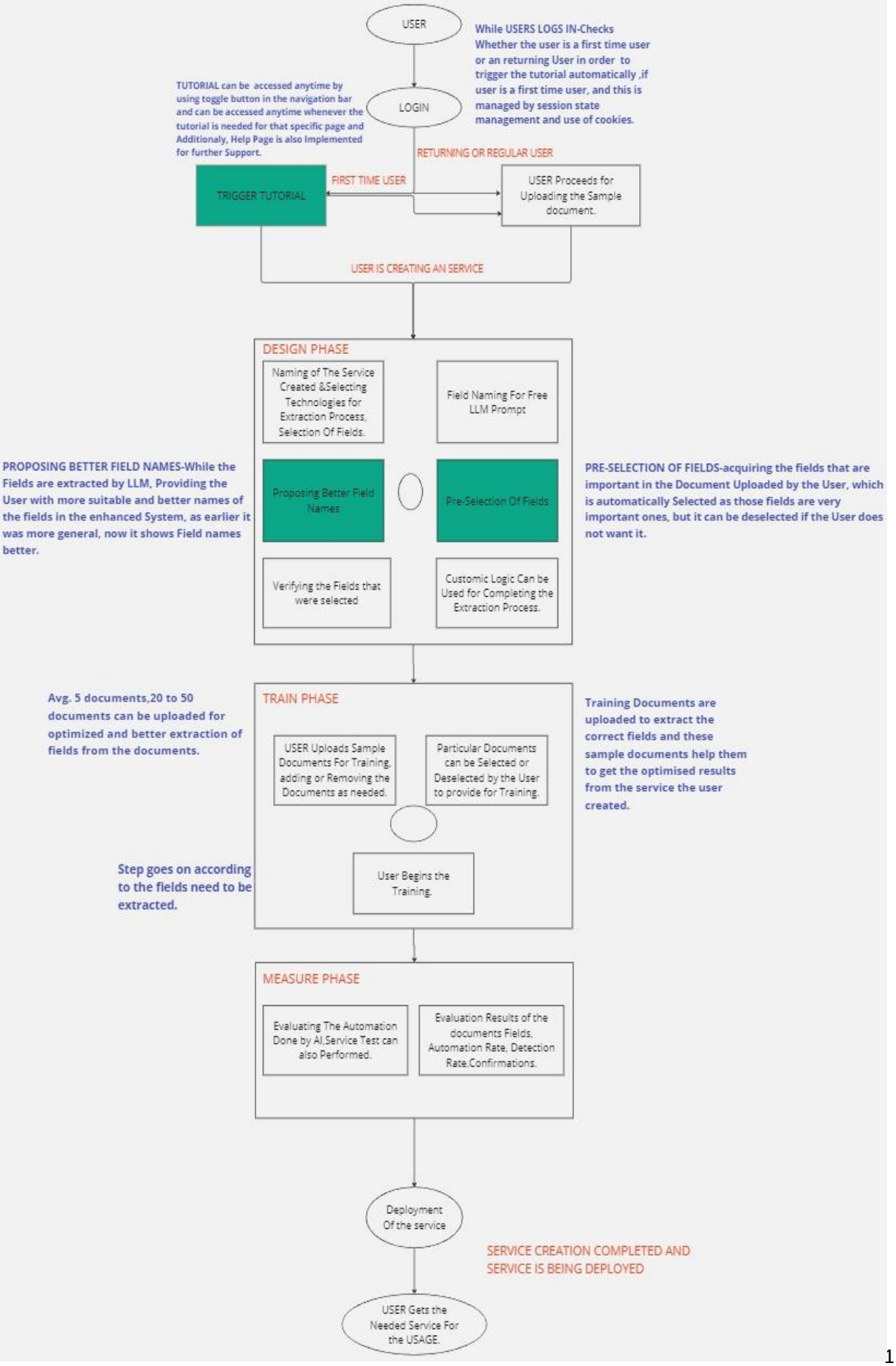
The system was initially designed with a structured flow for service creation, encompassing several key phases. Users begin by logging in and uploading sample documents to initiate service creation. In the **Design Phase**, users name the service, select technologies for data extraction, define field names, and verify selections, with the option to use custom logic for enhanced extraction. The **Train Phase** allows users to upload, add, or remove training documents, and select relevant files for training, culminating in the training process. The **Measure Phase** involves evaluating the automated processes, performing service tests, and reviewing results to ensure accuracy. Once all phases are complete, the service is deployed, providing users with a ready-to-use, custom-built system. This comprehensive structure served as the foundation for the system we received initially.


USER
LOGIN
USER Proceeds for Uploading the Sample document.
USER IS CREATING AN SERVICE
DESIGN PHASE
Naming of The Service Created &Selecting Technologies for Extraction Process, Selection Of Fields.
Field Naming For Free LLM Prompt
Verifying the Fields that were selected
Customic Logic Can be Used for Completing the Extraction Process.
TRAIN PHASE
USER Uploads Sample Documents For Training, adding or Removing the Documents as needed.
Particular Documents can be Selected or Deselected by the User to provide for Training.
User Begins the Training.
MEASURE PHASE
Evaluating The Automation Done by AI,Service Test can also Performed.
Evaluation Results
SERVICE CREATION COMPLETED AND SERVICE IS BEING DEPLOYED
Deployment Of the service
USER Gets the Needed Service For the USAGE.

3.1.2 Enhanced System

The enhanced system builds upon the initial framework with significant improvements for user experience and functionality, as shown in the flow. A tutorial feature (triggered automatically for first-time users via session state management and cookies) was implemented, with an option for users to access it anytime via a toggle in the navigation bar or a dedicated help page for additional support. In the **Design Phase**, two key functionalities were introduced: *Proposing Better Field Names*, where the system suggests more suitable and specific field names for better clarity, and *Pre-Selection of Fields*, where important fields are automatically identified and selected from uploaded documents but can be deselected by the user if not needed. The training process now accommodates 5 to 50 documents for optimal field extraction, and the **Train Phase** emphasizes step-by-step extraction according to field requirements. The **Measure Phase** evaluates automation using metrics like detection rate and confirmations. These enhancements, indicated in green, reflect functionalities implemented by us, addressing user needs and improving system accuracy and usability.

ENHANCED SYSTEM



1. User Interaction Layer

This layer is intended to oversee all user-facing interactions, such as service navigation, tutorials, and registration activities.

- **First-Time Users**: They are automatically directed through a comprehensive induction tutorial. This tutorial provides users with a comprehensive understanding of the system's functionality by guiding them through each phase.
- **Returning Users**: Maintain the ability to bypass the onboarding tutorial and directly engage in service creation or management duties. The tutorial state is managed using cookies to facilitate device-level recognition, thereby facilitating the provision of consistent and personalized user experiences.

2. Service Design Module

The **Design Phase** involves the configuration and customization of the extraction service:

- **Service Naming and Model Selection**: Users name the service and select a preferred LLM model for data extraction.
- **Field Configuration**:
  - Specify fields required for extraction.
  - Add custom logic for advanced extraction needs.
- **Field Review and Validation**: Ensures that the selected configuration aligns with the user's requirements before proceeding.

3. Training Module

The **Train Phase** ensures the accuracy and reliability of the extraction:

- **Document Upload and Curation**:
  - Users upload 5–50 sample documents for training.
  - Document addition or removal is allowed to optimize training results.
- **Training Process**:
  - Users confirm extracted field values to validate the system's accuracy.
  - The system learns to fine-tune its extraction process based on user feedback.
- **Iterative Workflow**:
  - Users can iteratively upload and validate new documents until desired accuracy is achieved.

4.Measure Phase (Evaluation and Testing)

The **Measure Phase** involves assessing the system's performance in extracting data from documents. This phase ensures that the automation delivers accurate results, aligns with user requirements, and performs reliably. Below is a step-by-step explanation based on the flowchart:

1. Evaluate Automation Done by AI

- After the training phase is completed, the system evaluates the extraction performance:
    - It uses the uploaded documents and field mappings from the training phase.
    - Automated extraction outputs are reviewed to identify discrepancies or errors.
    - Metrics like **field accuracy, automation rate, and detection rate** are calculated to measure performance.
- Users can review these evaluations to confirm the quality of the AI automation.

2. Perform Service Test

- A **service test** is conducted to simulate real-world usage.
- This involves:
    - Running the extraction service on unseen or test documents.
    - Ensuring that the fields are extracted accurately as per the configurations set in the design phase.
- The test helps verify the system's readiness before deployment.

3. Evaluate Results

- Post-testing, a detailed evaluation is performed, focusing on:
    - **Field Extraction Accuracy**: Ensures all required fields are captured correctly.
    - **Automation Rate**: The percentage of tasks the system performs without human intervention.
    - **Detection Rate**: How well the system identifies the intended fields.
    - **User Confirmations**: Any manual corrections or confirmations done by the user are reviewed.
- The results from this evaluation provide the user with clear insights into the system's reliability and efficiency.

4.Transition to Deployment

Once the evaluation metrics meet the user's expectations, the process moves to the deployment phase, where the service is finalized for operational use. This ensures the system is fully optimized and reliable before it begins handling production-level tasks.

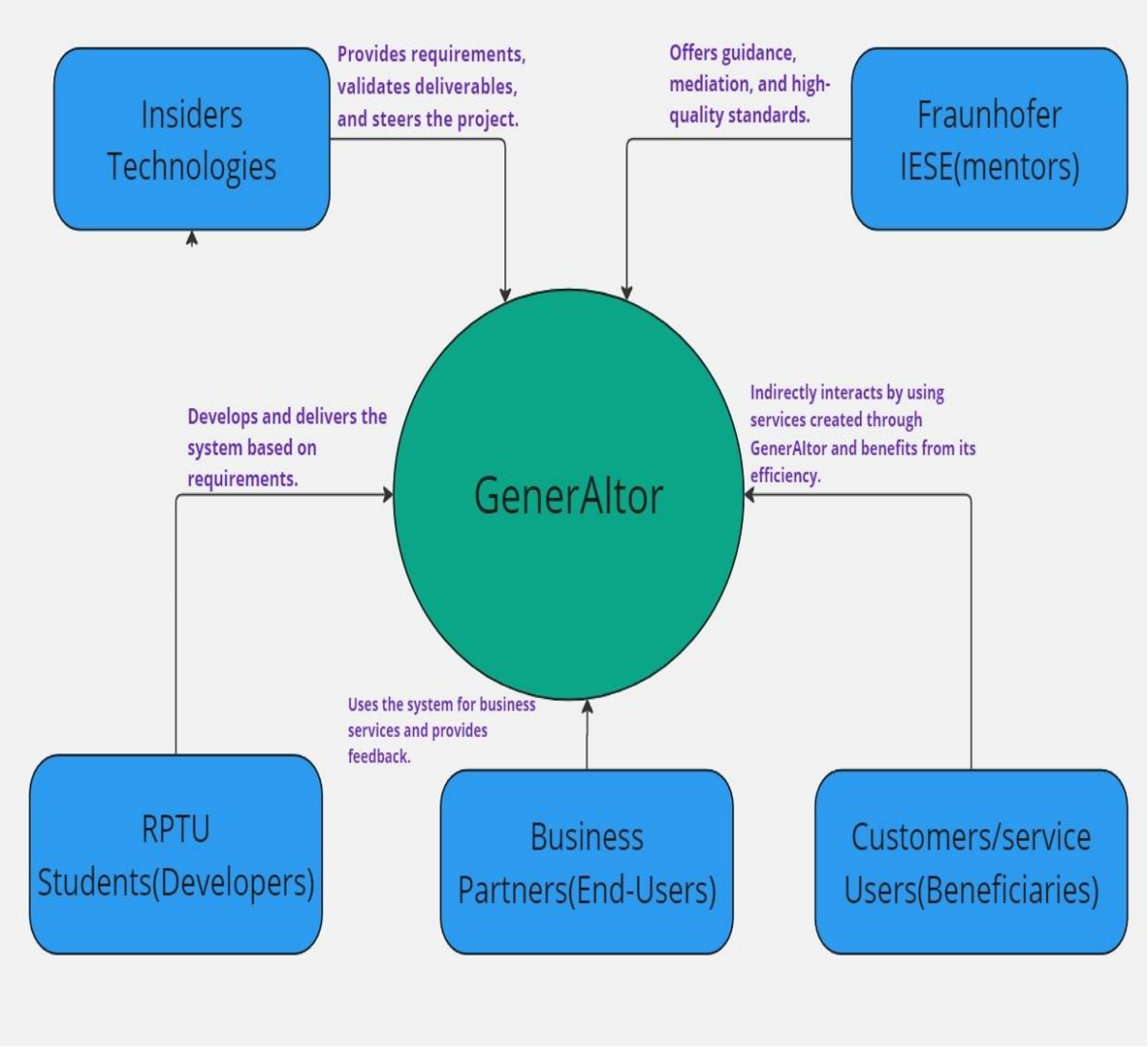
5.Deployment and Delivery Module

The **Deployment Phase** is the final step:

- After successful validation and testing, the configured service is deployed for regular use.
- Users access the service to perform real-world document extractions.

3.1.3 System Context DIAGRAM-(STAKEHOLDERS)

Insiders Technologies, providing requirements, verifies deliverables, and guides the project, and Fraunhofer IESE (mentors), who provide direction, mediate disputes, and guarantee high standards, are among GenerAltor's stakeholders. Based on these specifications, RPTU students (developers) are in charge of creating and deploying the system. End users, or business partners, use the system for business services and offer suggestions to enhance its performance. Lastly, by interacting with the results, customers/service users (beneficiaries) gain an indirect advantage from the effectiveness of the services produced by GenerAltor. These connections guarantee a cooperative effort to successfully develop and improve the system.



3.2 Key Architecture Decisions

First, we tested Shepherd.js and several other open-source JavaScript libraries for developing the tutorial functionality. However, we chose to move forward with Intro.js due to technical limitations in integrating Shepherd.js with the current system. A reliable and adaptable solution that could be tailored to match customer requirements was provided by Intro.js. Although we thought about creating the feature from the ground up, Intro.js was finally selected because of its user-friendliness, simplicity of usage, and compatibility with the team's experience, which guaranteed effective development and maintainability.

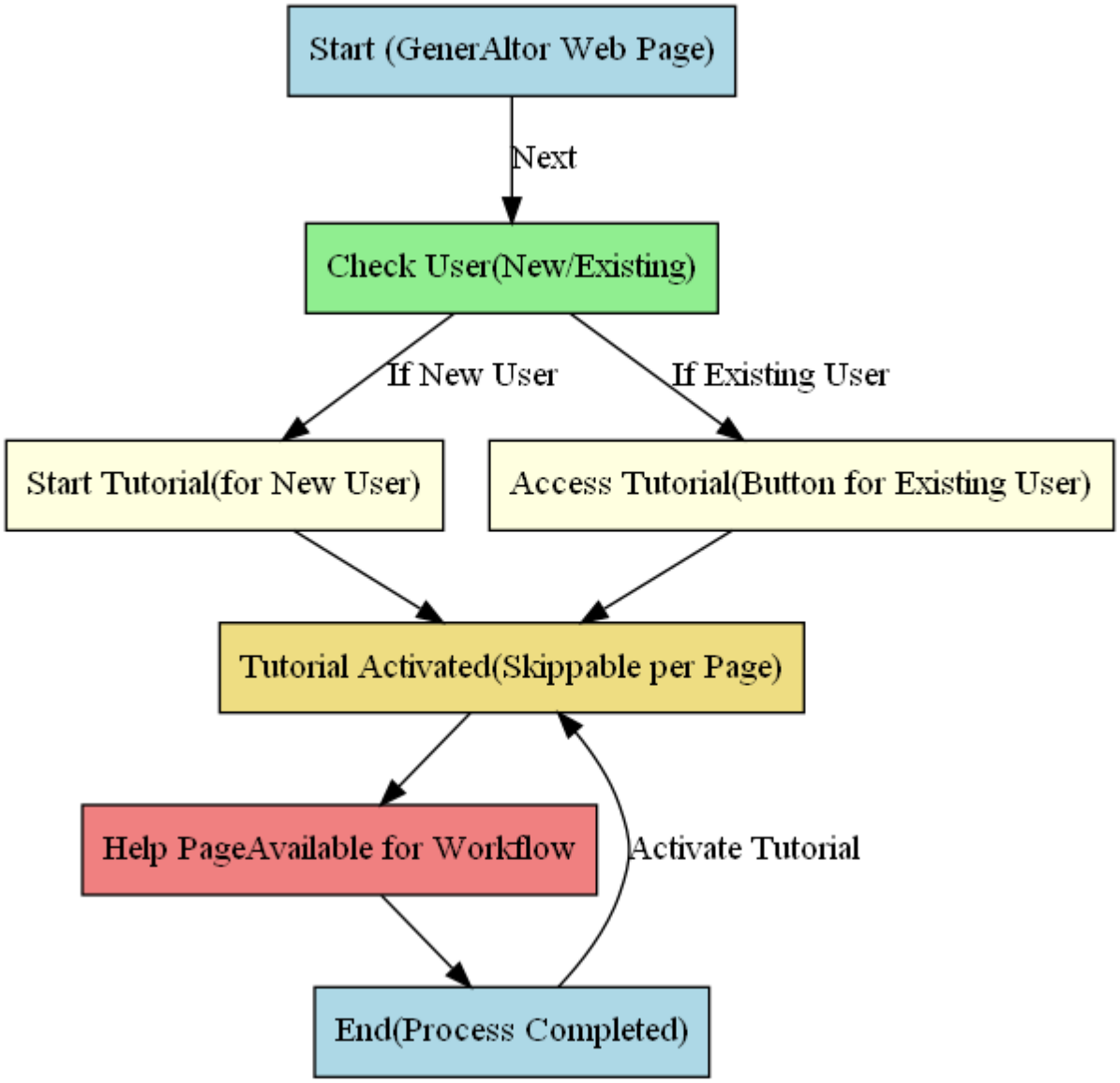
Using cookies to control the instructional state instead of keeping it on the server was another crucial architectural choice. With this method, the tutorial functionality is guaranteed to function per-device and provide instant feedback without necessitating server-side interactions. Cookies make it possible to differentiate between new and returning users in a lightweight manner, which makes implementation easier and lowers server demand. If the state had been kept on the server, account-specific management would have been required, which would have complicated the backend and required user sessions to be authenticated. Because cookies eliminate the need for a server-based user profile, both authorized and non-authenticated users can be onboarded with ease. This choice maximizes efficiency and supports the objective of offering a seamless and accessible user experience, even though it means that the tutorial state is not synced between devices.

In the backend architecture for enhancing field naming and pre-selection, key decisions include leveraging a multi-step process that integrates OCR data and advanced machine learning capabilities. For proposing better field names, the system retrieves document OCR artifacts, identifies selective fields, and uses machine learning models to analyze document context and refine field names with enhanced relevance and user-friendliness. This process ensures contextual accuracy by feeding both document context and specific field data into the analysis pipeline. For field pre-selection, a structured workflow extracts proposed fields, filters critical data points, and reintegrates these into the original structure through API calls.

3.3 Solution Approach

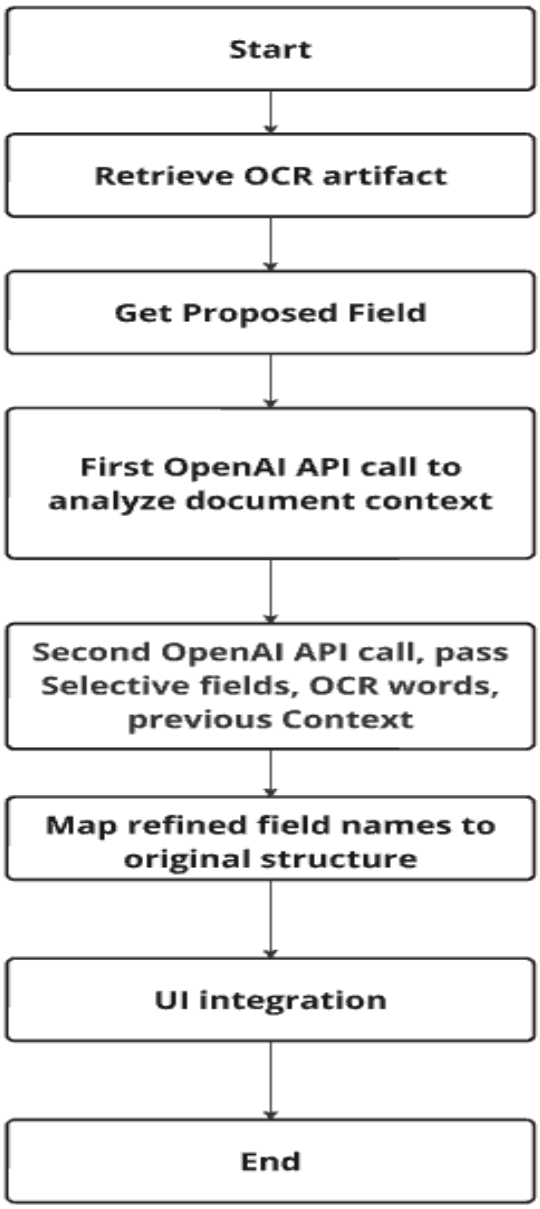
3.3.1 Frontend workflow

The goal of the GenerAItor tutorial is to improve user support and onboarding. When the GenerAItor page for document information extraction loads, the system uses cookies to determine if the user is returning or new. The tutorial starts immediately for new users and walks them through the process of creating a service step-by-step. A dedicated ‘Tutorial’ button allows returning users to access the instruction. Every page has the tutorial engaged to provide thorough help, but users can choose to skip it for the current page and reactivate it at the next workflow stage. Furthermore, a help page is always accessible, providing thorough instructions and answers to frequently asked questions, guaranteeing that users can easily explore and finish tasks in GenerAItor.



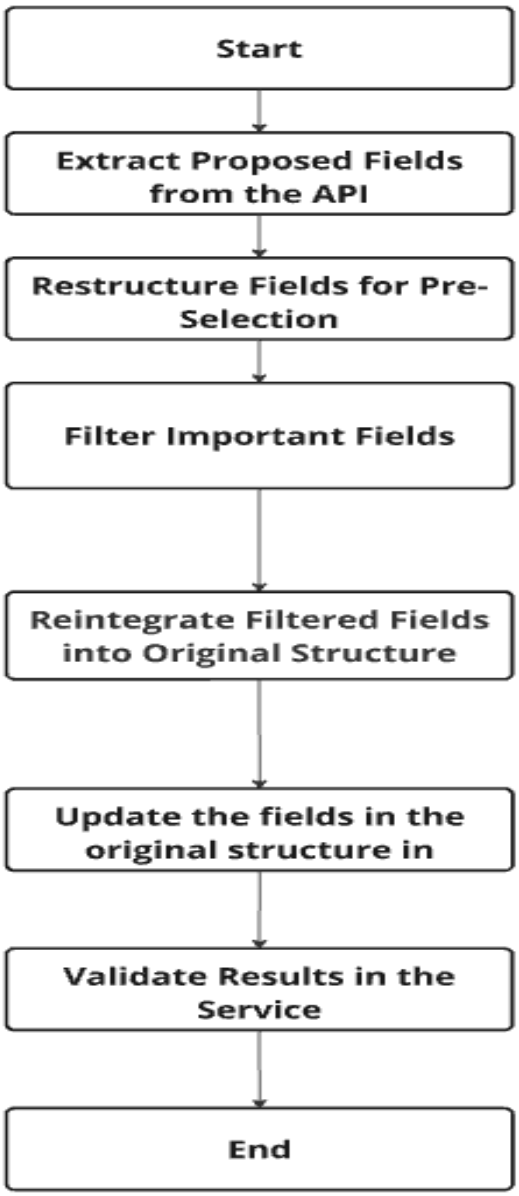
3.3.2 Proposing better field names

The workflow involves a systematic document processing approach that begins with retrieving document OCR and selective field data, followed by two strategic OpenAI API calls to analyze and contextualize the document. During the first API call, the system examines the document's context, and the second call integrates selective fields, OCR words, and the previously generated context. The process completes by mapping refined field names back to the original document structure and concluding with UI integration, creating a comprehensive method for intelligent document analysis and field extraction.



3.3.3 Flowchart for Pre-Selection of the fields

The workflow is a structured process for handling document fields, beginning with API-based field extraction and progressing through systematic refinement. After retrieving proposed fields, the system restructures and filters them to identify the most important data points. These filtered fields are then carefully reintegrated into the original document structure, ensuring data integrity. The workflow culminates by updating the defined fields through a change request and conducting comprehensive service-level validation, ultimately transforming raw field data into a precise, streamlined set of meaningful information.



3.4 Technologies Overview

3.4.1 Front-End Technologies

1. **Intro.js**:
    a. A lightweight, open-source library used to create step-by-step guides and product tours on websites or web applications.
    b. Helps improve user onboarding by providing interactive tutorials directly within the application interface.
2. **Node.js**:

a. A runtime environment that enables the execution of JavaScript code on the server side.
b. Known for its non-blocking, event-driven architecture, it's widely used for building scalable and high-performance web applications.

3.4.2 Back-End Technologies

1. **REST APIs**:
    a. Representational State Transfer (REST) APIs are a set of rules that enable the communication between the client and server over HTTP.
    b. They are stateless and typically use JSON for data exchange, making them lightweight and ideal for web-based applications.
2. **Kotlin**:
    a. A modern, concise programming language that runs on the Java Virtual Machine (JVM).
    b. Known for its interoperability with Java, it's frequently used for Android app development and creating backend services in frameworks like Spring Boot.
3. **Creation of Services**:
    a. Refers to the process of designing modular and reusable backend components that handle specific functionalities, such as user authentication, data processing, or integration with third-party services.
    b. These services are often built as microservices to ensure scalability and ease of maintenance.

4. Risks mitigations and Technical Debt

4.1 Risks mitigations

Unclear Requirements or Scope Changes

- *Description:* Incomplete or changing requirements can lead to misaligned development efforts and delays.
- *Mitigation:* Conduct regular stakeholder meetings, maintain a detailed requirements document, and implement a change management process.

Limited Team Expertise

- *Description*: The team's limited experience with specific technologies (e.g., OCR or API integration) may slow development and increase errors.
- *Mitigation:* Allocate time for skill development, consult experts, and conduct code reviews to ensure quality.

Integration Challenges

- *Description:* Difficulty integrating components such as OCR tools or AI APIs may disrupt workflows and delay timelines.
- *Mitigation:* Develop and test integration modules early, use mock APIs, and allocate buffer time for troubleshooting.

Unforeseen Technical Constraints

- *Description:* Technology stack limitations may restrict certain planned functionalities.
- *Mitigation:* Regularly evaluate technical feasibility and explore alternative tools or approaches.

4.2 Technical Debt

Temporary Workarounds

- *Description:* Implementing quick fixes instead of optimal solutions during tight deadlines may reduce code quality.
- *Impact:* Increased maintenance effort and potential rework in the future.
- *Plan to Address:* Document workarounds, allocate time for refactoring, and prioritize resolution in future sprints.

Incomplete Test Coverage

- *Description:* Limited testing for components may lead to undetected bugs.
- Impact: Higher risk of failures in production.
- *Plan to Address:* Gradually increase test coverage, starting with critical functionalities.

Code Duplication

- *Description:* Repeated code patterns may arise from rushed implementations or unclear modularization.
- *Impact:* Reduced maintainability and increased risk of inconsistent behavior.
- *Plan to Address:* Refactor code to reduce duplication and adhere to DRY (Don't Repeat Yourself) principles.

Incomplete Documentation

- *Description:* Hasty development may result in gaps in technical and user documentation.
- *Impact:* Steeper learning curve for new team members and users.
- *Plan to Address:* Establish documentation standards and allocate time for documentation in future iterations.

5. Glossary

- **API (Application Programming Interface):** A set of endpoints that allow external systems to interact with GenerAItor, facilitating document submission and data retrieval.
- **User:** Anyone who interacts with GenerAItor, including customers and business partners.
- **Business Partner:** A user responsible for configuring and tailoring services within GenerAItor to meet specific business needs.
- **Customer:** A user who leverages GenerAItor's services for efficient document data extraction and processing.

- **Data Fields:** Specific pieces of information extracted from a document, such as names, dates, or amounts.
- **Document:** The original file uploaded by the user, containing metadata such as type, format, and structure, to be analyzed by GenerAItor.
- **Field Group:** A collection of extracted fields grouped together to manage complex data structures effectively.
- **Field Name Proposals:** Suggested names for extracted fields, refined for accuracy and relevance.
- **Field Preselection:** The automated process of identifying and selecting the most relevant fields in a document before extraction.
- **JSON (JavaScript Object Notation):** A human-readable text-based format used to exchange data between web clients and servers.
- **Manual Validation:** The process where clients review and confirm the field names or values suggested by GenerAItor.
- **OCR (Optical Character Recognition):** Technology that converts images of text into machine-readable text, enabling the extraction of data from scanned documents or images.
- **Preselection:** The system's process of selecting key fields from a document based on its type and content before full extraction.

6. Conclusion

The Software Architecture Document (SAD) for GenerAItor provides a detailed blueprint for creating an advanced AI-powered document analysis tool. It emphasizes usability, scalability, and automation to deliver a user-friendly and efficient solution for document data extraction.

The SAD outlines innovative features like interactive tutorials, intelligent field preselection, and logical data organization, ensuring a seamless user experience while minimizing manual effort. By addressing quality attributes such as reliability, performance, and maintainability, it ensures the system’s effectiveness and adaptability for future enhancements.

This document serves as a strategic and technical guide for stakeholders and developers, ensuring the successful implementation of GenerAItor to meet real-world needs through innovative, AI-driven automation.