Synthesis project Architecture Description

Table of Contents

Introduction

This document describes solution proposal for the system which supports the following client business processes:

1. Sales Engagement Process SOW Generation (SOW)
2. Investigational New Drug FDA Submission Process (IND)

Goal of the system is to help the client to track and manage the listed business processes and to automatically generate the processes output artifacts, i.e. final documents that come out of these processes.

Definition of scope and key requirements will be documented outside of this document. Nevertheless, coarse, and probably incomplete, list of captured use cases can be found in appendix in the Requirements section.

For the quick overview of the solution please take a look at Context View section. More detail description of the system and its key functional elements can be found in the Functional View section.

Context View

This section of the document provides a high-level view of the system and its environment, showing how the system interacts with external entities. We will use following model to document the system's boundaries and identify its key relationships with users, other systems, and external environments.

Context model

Model depicted below shows the system in the context of the cloud environment.

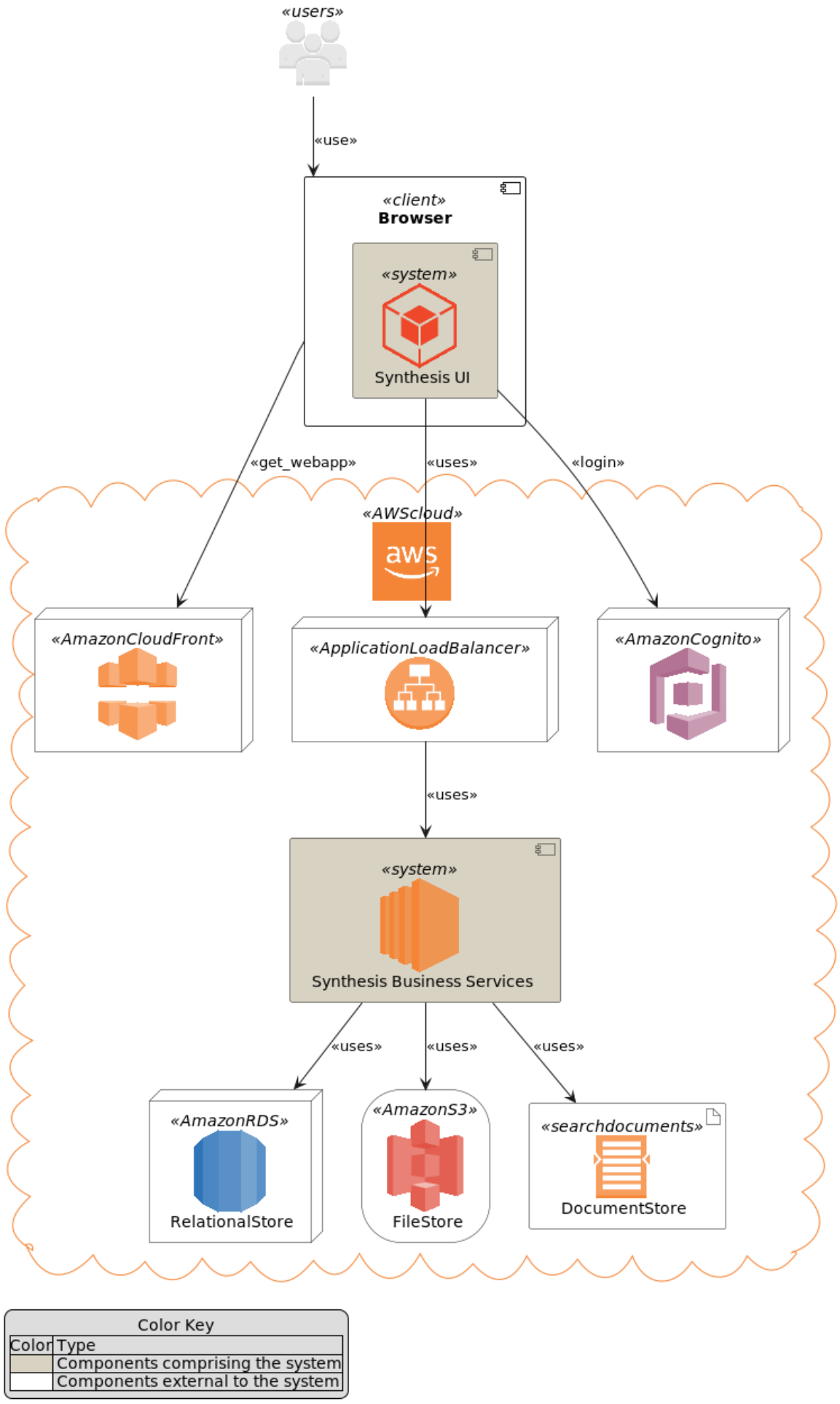


Figure 1. System context model

The system allows users to manage **Sales Engagement Process (SOW)** or **Investigational New Drug FDA Submission Process (IND)** through a web-based application.

The system has two main parts, the user facing web application (**Synthesis UI**) and the backend business services (**Synthesis Business Services**). The system interfaces with several external entities:

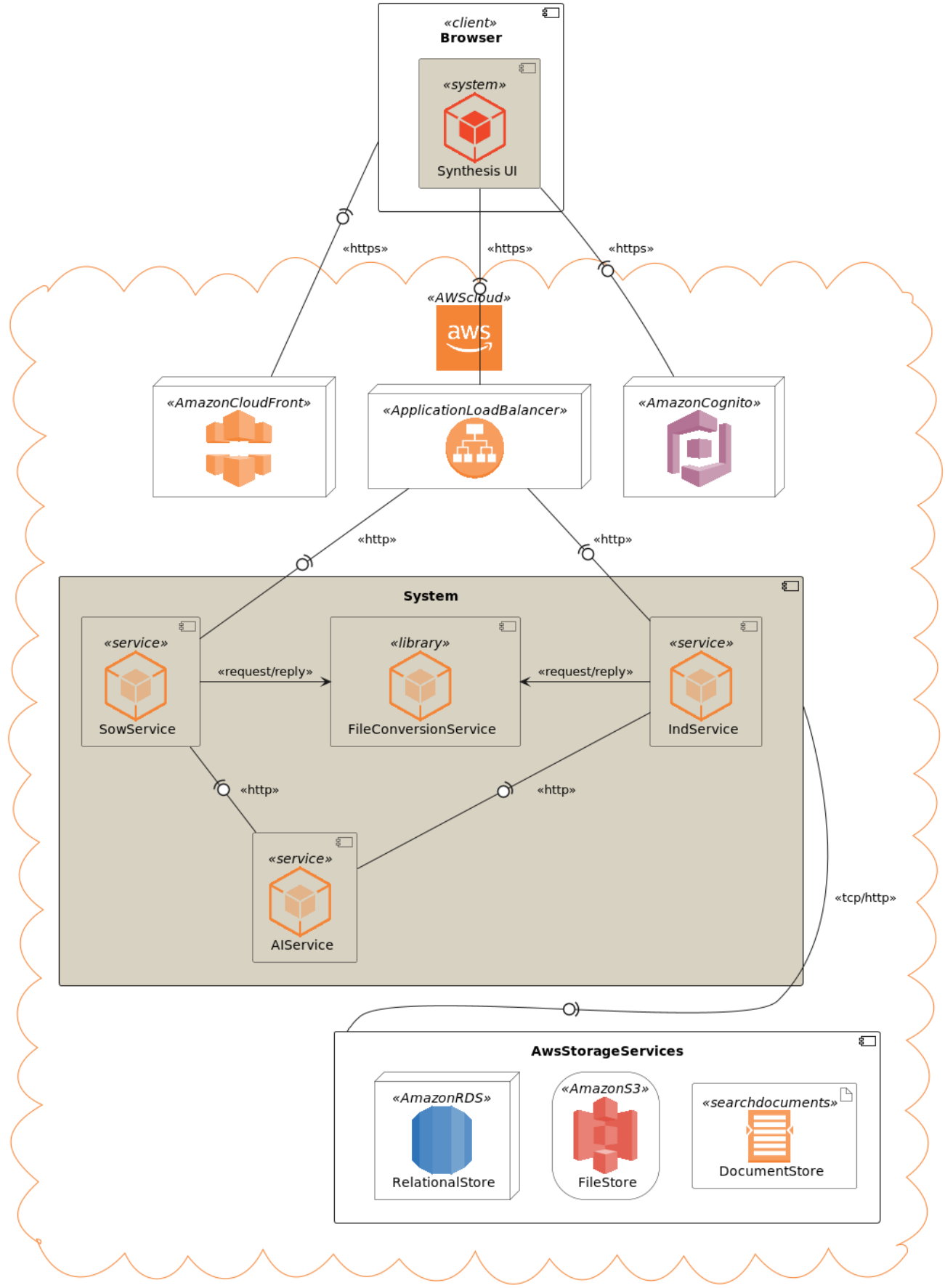
- **Users**: users, using Synthesis UI running in their browsers, log into the system to start new or manage existing SOW or IND processes. Users enter information specific to a process, upload files related to the process and generate output documentation.
- **Browser**: a software installed on user machines that runs Synthesis UI
- **AmazonCloudFront**: the service that hosts the Synthesis UI files. Browsers download these files and run them as Synthesis UI javascript application when users visit the system website.
- **AmazonCognitoService**: the service that hosts user details like username and password, and enables the Synthesis UI to acquire the security token needed for the communication with Synthesis Business Services
- **AmazonApiGateway**: the service that exposes the API of Synthesis Business Services to the internet, so it can be used by Synthesis UI. The service provides needed security and scalability features to enable Synthesis UI and Synthesis Business Services to communicate in a secure and scalable way.
- **AmazonRDS RelationalStore**: the service that provides relational database storage for the system data that needs to be persisted, typically metadata about entities managed by the system.
- **AmazonS3 FileStore**: the service that provides file storage the files that will be uploaded to the system or generated by the system.
- **AmazonS3 DocumentStore**: the service that store, retrieve, and manage document-oriented information for the system, typically in formats like JSON. This kind od format is suitable to be feed in large language models. Files from AmazonS3 FileStore will typically be converted into JSON and stored into this service.

Functional View

This section of the document describes system's functional structure, breaking it down into elements that deliver the functions of the system and detailing their responsibilities. We will define the system key runtime components, their responsibilities, the interfaces they expose, and the interaction between them.

Functional structure model

Model depicted below shows the system key functional runtime components.



Color Key	
Color	Type
	Components comprising the system
	Components external to the system

Figure 2. Functional model

The system is composed of five main functional components linked via a number of connector types. Because all system services are using some kind of storage, or more than a few, to reduce the clutter, model shows just one general tcp (or http in some cases) connector between the system and aws storage services. It should be clear from the description of the particular component, what storage services and connectors it is using.

- **SowService**: the component is responsible for management of SOW processes. The component supports creation of a new SOW process, editing the details of a process, upload and storage of files relevant to the process, generation of output documents. etc. The component provides REST API interface via HTTP request/reply connector to access the component functionalities. The component consumes a number of storage services provided by AWS, like:
  - RDS service to store and update metadata about each SOW process
  - S3 service to store files (either input files or generated files) relevant to a SOW process
  - DocumentDB service to keep textual, cleaned versions of the files stored in S3 service

  The component access all these storage services via well known and documented interfaces using either pure TCP or HTTP request/reply connectors. Documentation of the particular storage interfaces can be found on AWS site. The component consumes **AIService** to generate needed output documentation using **AIService** REST api via HTTP request/reply connector. The component embeds **FileConversionService** as a library. It uses its api via in process method calls to convert the SOW related files between different formats, like PDF to text and vice versa.

- **IndService**: the component is responsible for management of IND processes. The component is functionally similar to **SowService**, it just handles different business process. The component supports creation of a new IND process, editing the details of a process, upload and storage of files relevant to the process, generation of output documents. etc. The component provides REST API interface via HTTP request/reply connector to access the component functionalities. The component consumes a number of storage services provided by AWS, like:
  - RDS service to store and update metadata about each IND process
  - S3 service to store files (either input files or generated files) relevant to a IND process
  - DocumentDB service to keep textual, cleaned versions of the files stored in S3 service

  The component access all these storage services via well known and documented interfaces using either pure TCP or HTTP request/reply connectors. Documentation of the particular storage interfaces can be found on AWS site. The component consumes **AIService** to generate needed output documentation using **AIService** REST api via HTTP request/reply connector. The component embeds **FileConversionService** as a library. It uses its api via in process method calls to convert the IND related files between different formats, like PDF to text and vice versa.

- **AIService**: the component provides access to large language models specifically fine-tuned for generation of SOW and IND output documents. The component provides REST API interface via HTTP request/reply connector to access the component functionalities. Access to these services is done via well known and documented interfaces. Documentation of the particular service interface can be found on AWS site. **SowService** and **IndService** are consumers of this component.
- **FileConversionService**: the component provides functionality to convert files from one format

into another, e.g. from PDF into pure text or JSON format. The component is a utility library which can be imported into a particular service. The component provides file format conversion API which can be invoked via in process method call. **SowService** and **IndService** are consumers of this component.

- **Synthesis UI**: the component provides user interface for the system. The component is single page javascript application which runs in Users browser. Using this component, users can start new or manage existing SOW or IND processes. Users can enter information specific to a process, upload files related to the process and generate output documentation. The component consumes **SowService** and **IndService** REST api indirectly via **AmazonApiGateway** infrastructure component. This way the component can utilize secure HTTPS (HTTP over TLS) connector of the **AmazonApiGateway**. The **AmazonApiGateway** component will terminate TLS connection on it's end and forward the HTTP request to the appropriate internal component, either to **SowService** or **IndService**.

Appendices

Requirements

This section is a free interpretation of the requirements by the author of this document from information collected at the discovery calls held with the client.

IMPORTANT	Official requirements shall be captured elsewhere and documented outside of this document.

Sales Engagement Process SOW Generation requirements

Following picture denotes the use case model for the Sales Engagement Process envisioned and proposed by the author of this document.

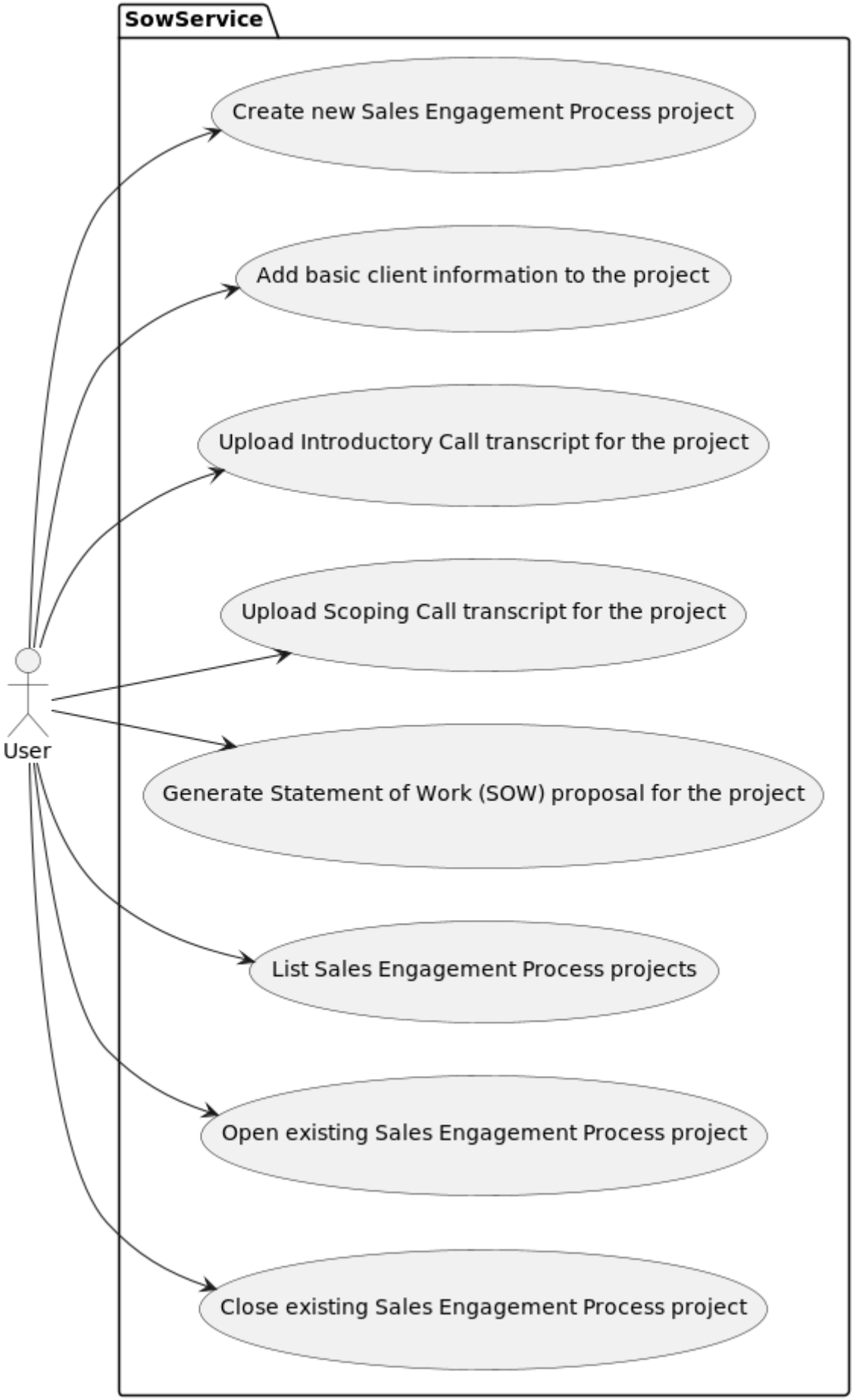


Figure 3. Sales Engagement Process SOW Generation Uses Cases

Investigational New Drug FDA Submission Process requirements

Following picture denotes the use case model for the Investigational New Drug FDA Submission Process envisioned and proposed by the author of this document.

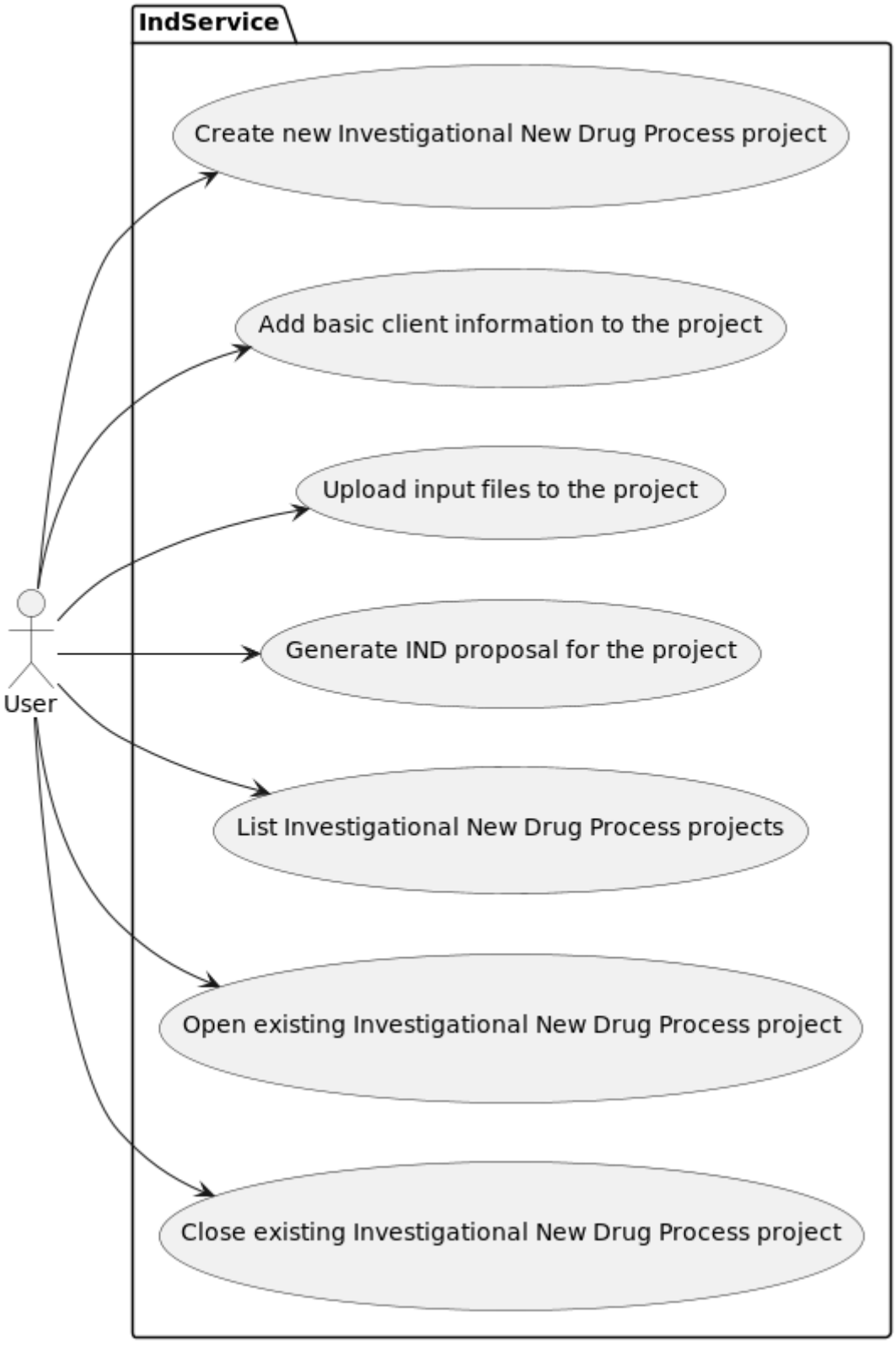


Figure 4. Investigational New Drug FDA Submission Process Uses Cases