# Design Document

Neolook Solutions
AI Switchboard

**TA:**

Ieva Randyte

**Team:**

Delia-Andreea Popa
Victor Andrei Didraga
Vasco Reynolds Brandao
Raluca-Alexia Neatu

*University of Groningen*

## Contents

## 1 Technology Stack

In this section, we will introduce the technologies/frameworks and programming languages used for developing this project.

### 1.1 MVP

The following diagram depicts the technologies used to develop the MVP for our application. For the backend and the database we have chosen Django, Python and mySQL, as it was recommended by the company to do so. For the integration of the AI models, it was suggested to us to test it using different AI models from the Hugging Face platform. As for the frontend, we also aimed to follow the company's preferences, and use JavaScript, but due to the short time frame that we had available for the MVP, we chose to keep the frontend in Django and HTML, and switch to JavaScript and CSS for the final product.

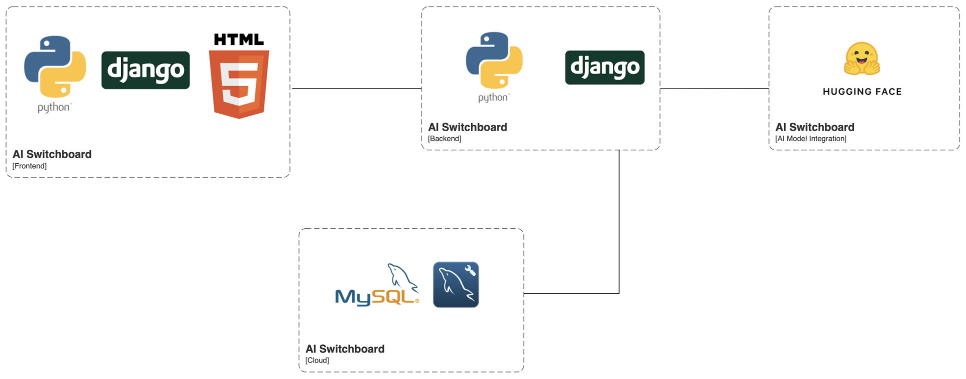

Figure 1: MVP Technology Stack

### 1.2 Final Product

As previously mentioned, the technologies used for the frontend were eventually changed to JavaScript, CSS and HTML for the final product, as it was recommended by the company to use them. The following diagram displays this change of technologies.

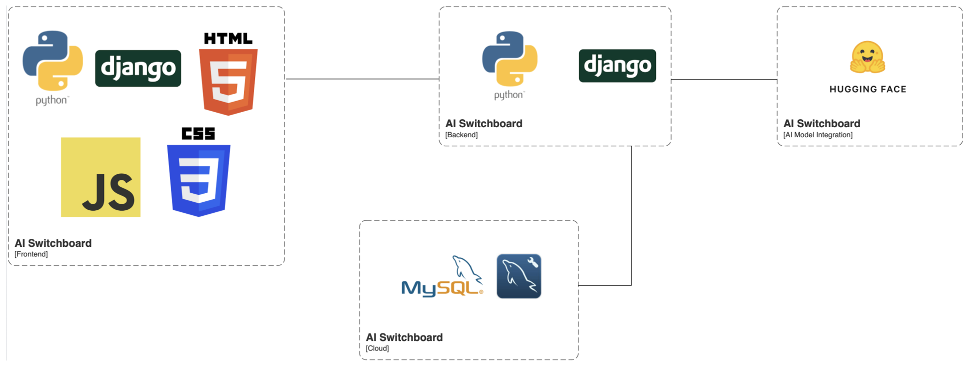

Figure 2: Final Product Technology Stack

## 2 Architecture

### 2.1 Architectural Overview - System Context

The System Context diagram shows the overall interaction between the user and the elements of the system. The user/medical staff only directly interacts with the web application, while the two external systems, the database and the model, interact back and forth with the application.

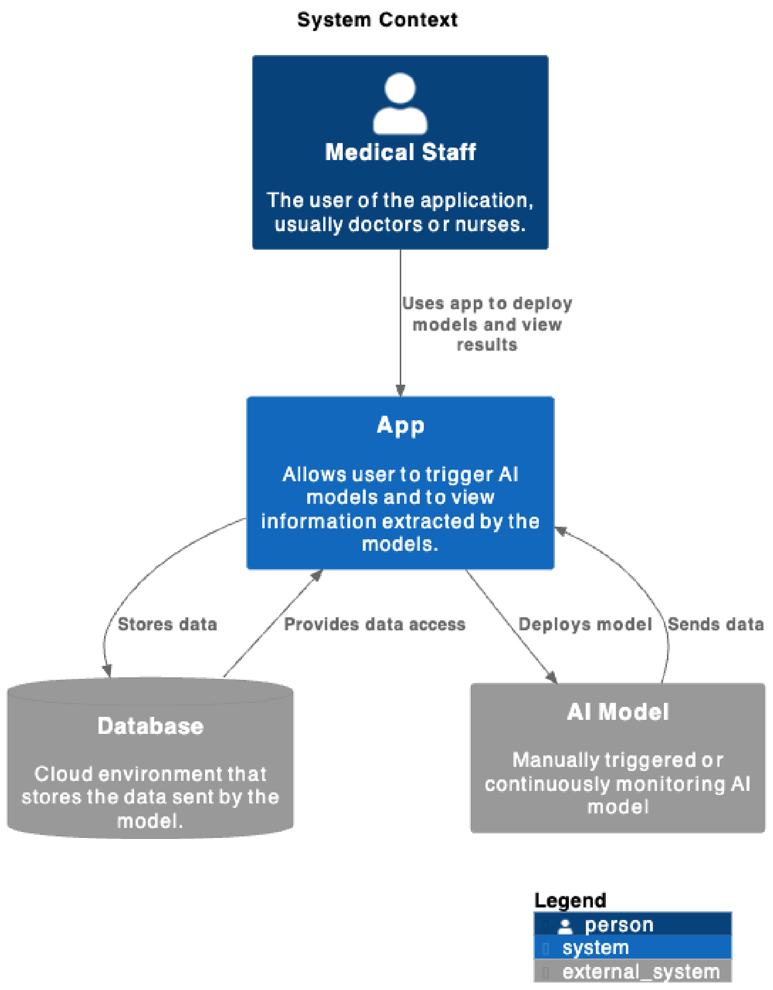
System Context
Medical Staff
The user of the application, usually doctors or nurses.
Uses app to deploy models and view results
App
Allows user to trigger AI models and to view information extracted by the models.
Stores data
Provides data access
Deploys model
Sends data
Database
Cloud environment that stores the data sent by the model.
AI Model
Manually triggered or continuously monitoring AI model
Legend
person
system
external_system

Figure 3: System Context Diagram

## 2.2 Architectural Overview - Container View

The Container View shows a further broken-down view of the application, which shows the interaction between its components, along with their responsibilities. In the following sections, we expand on each components' functionality.

There are two diagrams for the container view, since we use a different technology and application design for the MVP.

### 2.2.1 MVP

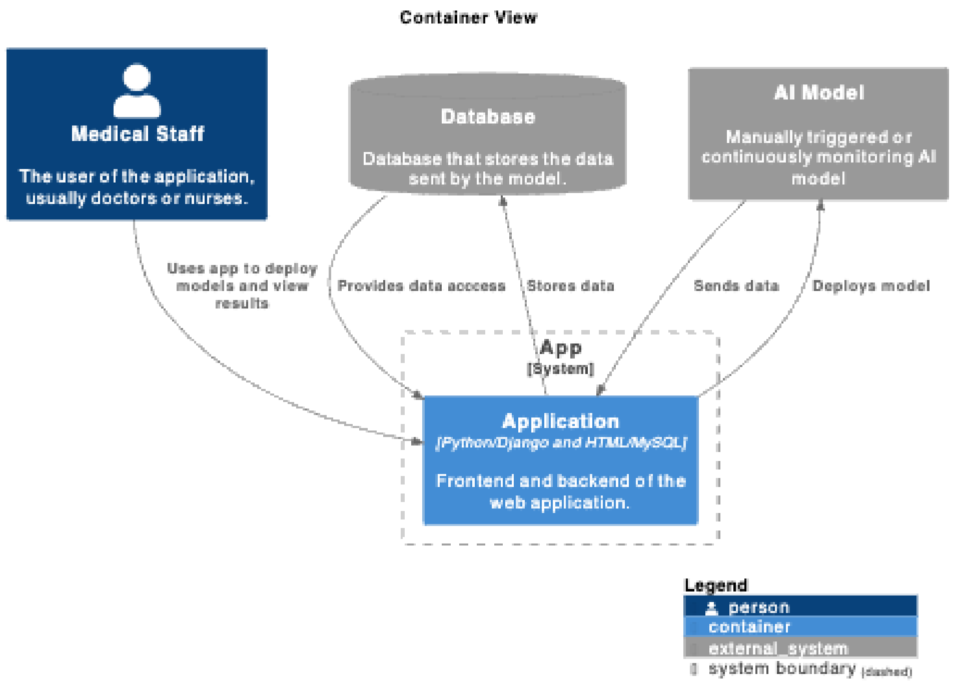
Container View
Medical Staff
The user of the application, usually doctors or nurses.
Database
Database that stores the data sent by the model.
AI Model
Manually triggered or continuously monitoring AI model
Uses app to deploy models and view results
Provides data access
Stores data
Sends data
Deploys model
App
[System]
Application
[Python/Django and HTML/MySQL]
Frontend and backend of the web application.
Legend
person
container
external_system
system boundary (dashed)

Figure 4: Container Diagram for the MVP

### 2.2.2 Final Product

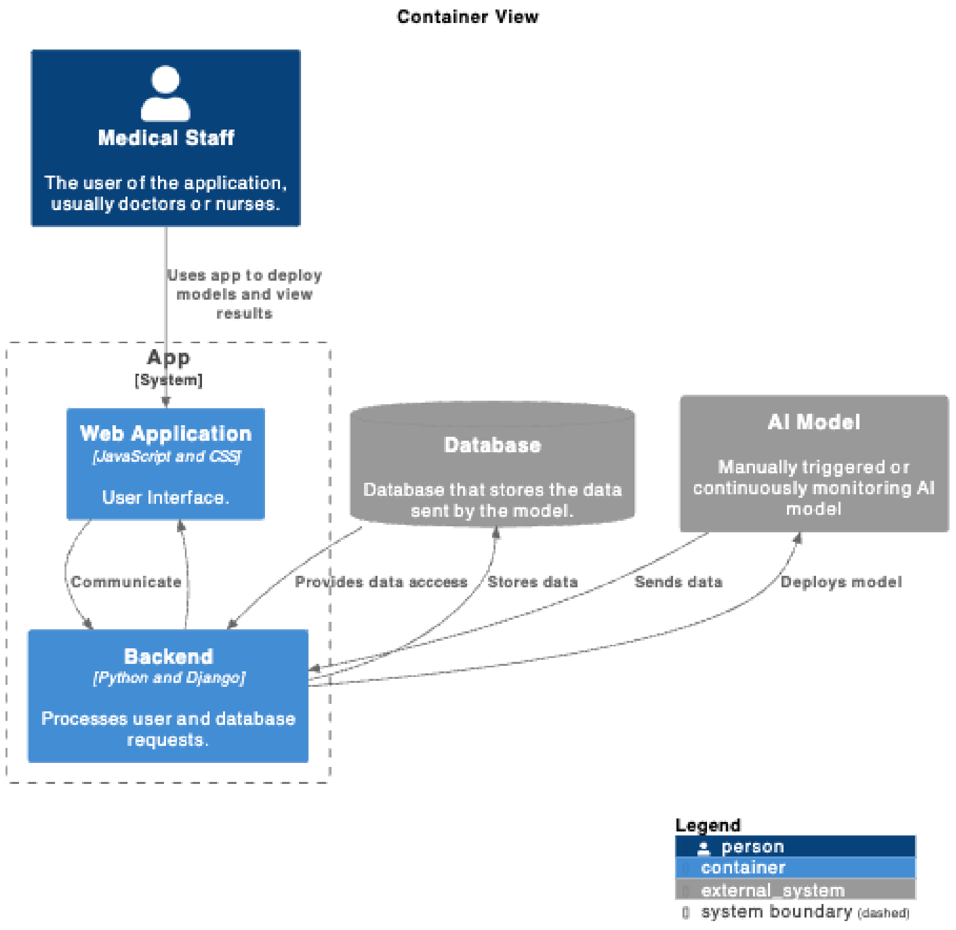

Figure 5: Container Diagram for the Final Product

## 2.3 Web Application

The Web Application is responsible for providing a proper User Interface to the medical staff, and appropriate backend logic that is able to process the user requests in order to produce corresponding outputs/responses. The UI/frontend is mainly used for the display of the output data and notifications, and for the selection of AI models. The backend is in charge of communicating with the frontend, in order to fulfill the requests sent by the medical staff, and with the database, in order to store AI model output and patient data. It is also used for fetching certain patient information, and for delivering the appropriate notifications generated by the models' output.

The AI models specified throughout this document are developed and provided by the company, and we only interact with them through an API. Therefore, there are no details provided in regard to their design in this document, since we do not have direct access to these models, and their actual functionality is not relevant to our project. For this project, we are only concerned with the way they interact with the other components, since our task is just integrating them inside a web application.

However, since the company refused us access to their models, in order for us to test our web application, we created two simple models that substitute the more sophisticated ones developed by the company. One of them is an edge-detection model that is supposed to act as a run-on-demand model for testing purposes, and the second one is a scream detection model that is supposed to substitute a continuous model.

In the following subsections, we focus on mapping the requirements mentioned in the previous document to the functionality of the Web Application.

### 2.3.1 Run-on-demand AI models

The interaction diagram presented in this section aims to fulfill the following requirements: M-RQ-F1,M-RQ-F2,M-RQ-F3,S-RQ-F4,S-RQ-F5,S-RQ-F6,C-RQ-F7,C-RQ-F8,C-RQ-F9,C-RQ-F10,M-RQ-F11. It encompasses the selection of a specific AI model, the uploading of data/fetching data from the database as input for AI models and the ability to download the output given by the AI models.

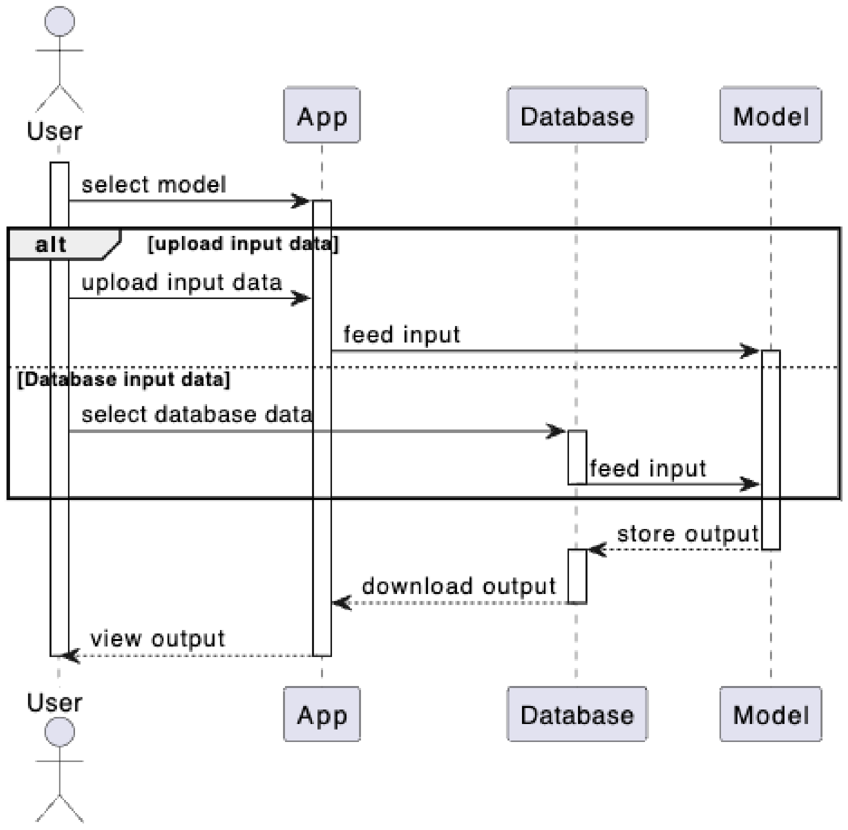

Figure 6: Interaction Diagram for Run-on-demand AI models

### 2.3.2 Continuous AI models

This section focuses on giving an insight on the interaction diagrams created for the requirements targeted at the continuous AI models.

**Switching on/off the continuous models** This interaction diagram depicts the process of switching on and off the continuous running models from the web application. At first, the medical staff interacts with the frontend interface, where a specific button is provided to open a table which displays the available continuous models. For each of the models, there is a button which allows the user to switch them on or off.

The following diagram aims to fulfill the following requirements: C-RQ-F13, C-RQ-F14.

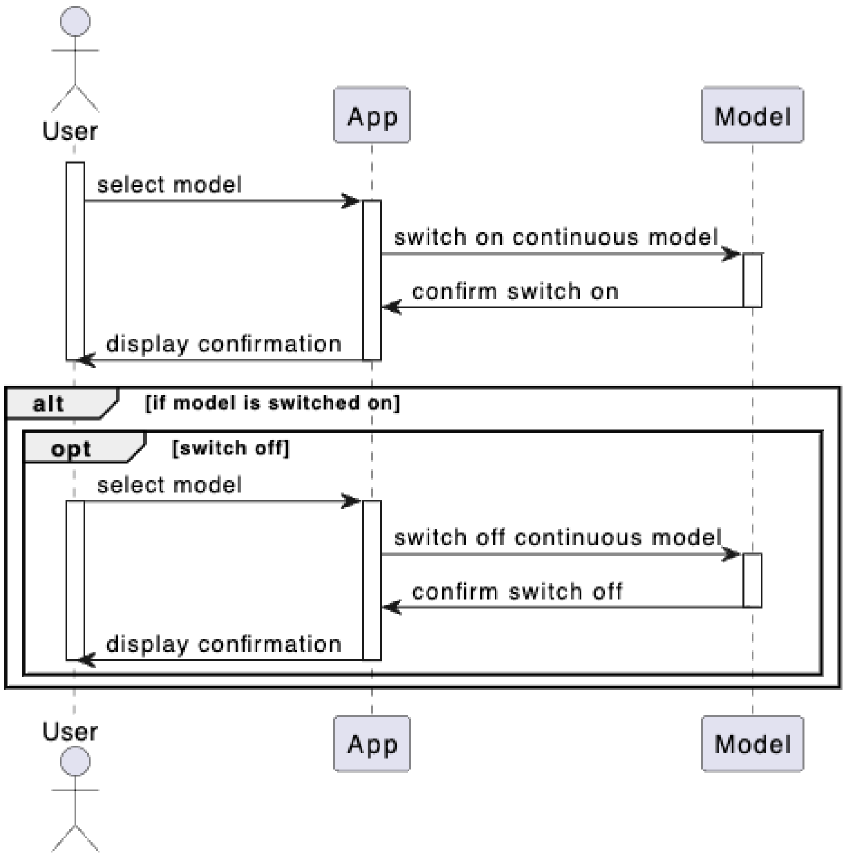

Figure 7: Interaction Diagram - on/off process for the Continuous AI models

**Automated Notification System** This interaction diagram depicts the process of the staff receiving notifications from the continuous running models, in case unusual data is identified by the model. Assuming that the model is already running and posting the data continuously to the database, in case unusual data is posted to the database, a message is sent to the Web Application. This message notifies the user that abnormal activity was detected, and provides the data which triggered the alert. Moreover, the Web Application provides the user with a button that allows them to download specific output data that triggered the entire notification process. The data is then displayed based on the user's choice.

This diagram aims to fulfill the following requirements: M-RQ-F12, M-RQ-

F15, C-RQ-F16, S-RQ-F17, S-RQ-F18, S-RQ-F19, S-RQ-F20, S-RQ-F21, M-RQ-F22, M-RQ-F23, C-RQ-24, C-RQ-25.

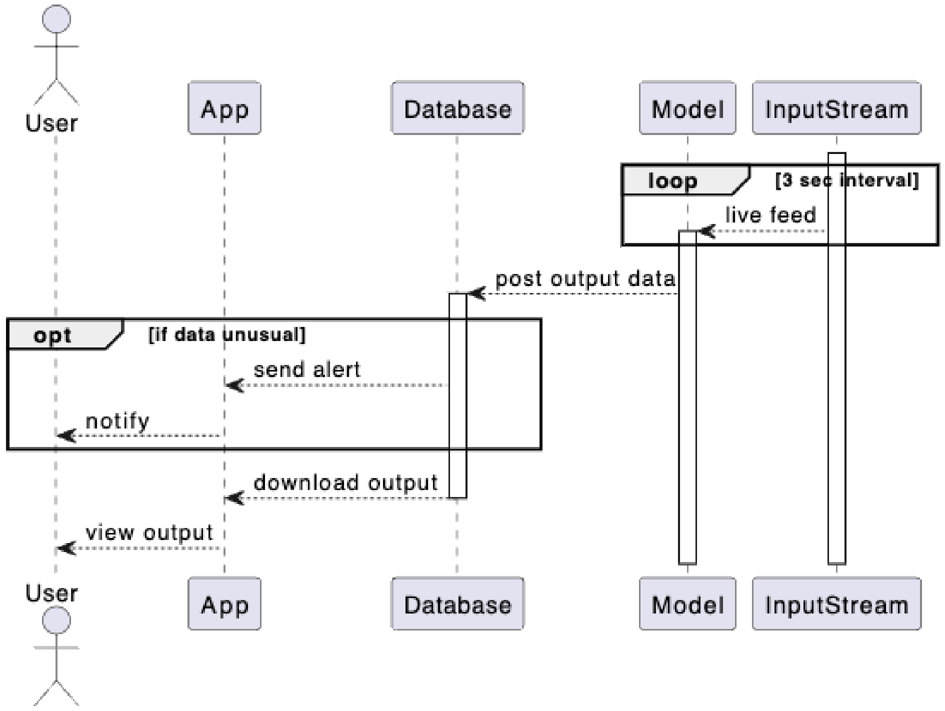

Figure 8: Interaction Diagram for the Automated Notification System

## 2.4 Database

The database represents the database of the application. In this environment is where the AI models' output is going to be stored, along with additional data. Additionally, we list the interaction diagrams that align with the requirements regarding the database used for the application.

### 2.4.1 Retrieving patient data

The following interaction diagram depicts the process by which the patient data can be downloaded from the database via a button on the frontend interface. Initially, the medical staff interacts with the frontend interface, where a specific button is provided to start the download process and select the data to be downloaded. Upon clicking this button, a request is generated and sent from the frontend to the backend, which forwards it to the database. The data is fetched and sent back to the frontend through the backend, where it is converted in the type of file based on the user's format choice(plain text/csv/JSON).

This diagram aims to fulfill the following requirement: S-RQ-F26, S-RQ-F27.

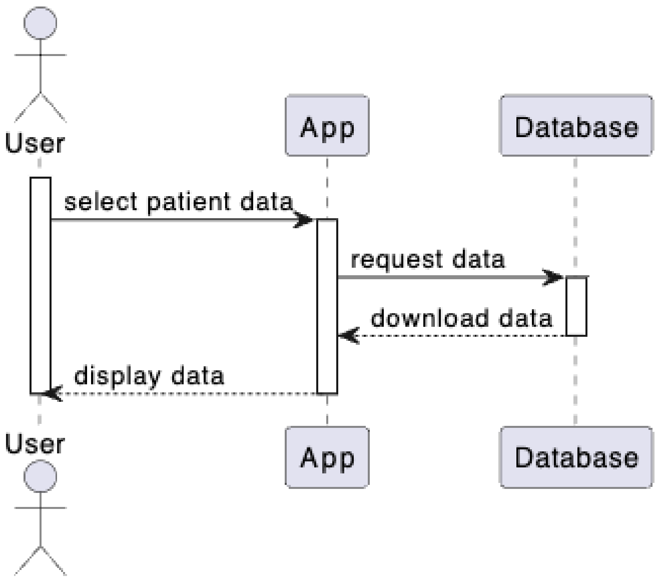

Figure 9: Interaction Diagram for retrieving patient data from the database

### 2.4.2 Storing patient data

The interaction diagram portrays the process for storing patient data in the database through a user-initiated action on the frontend. The medical staff interacts with the frontend interface, providing a file with the patient data to be stored, initiating the transfer of it to the database. This transfer involves communication between the frontend interface and the database through the backend, ensuring seamless transmission and storage of the information. After this process is done, a confirmation is transferred to the frontend interface, notifying the user that the data has been saved successfully.

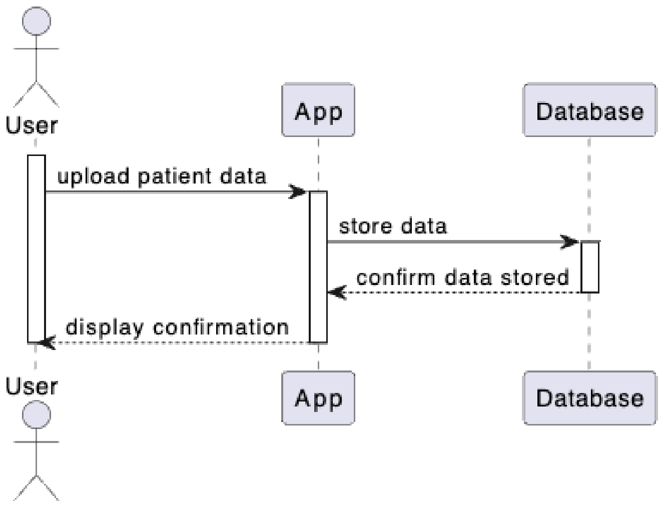

Figure 10: Interaction Diagram for storing patient data in the database

This diagram aims to fulfill the following requirements: M-RQ-F28, M-RQ-F29, S-RQ-F30.