|  |
| --- |
|  |
| Design Document |
|  |

|  |
| --- |
| Group: Team Bae  Members: Ralph, Martijn, Max, Lars en Gijs  Class: S3 DB-03  Tutor: Mark Mestrom  Date: 27-11-2023 |

Contents

[Layer Diagram 2](#_Toc151988676)

[C4 model 3](#_Toc151988677)

[Level 2 3](#_Toc151988678)

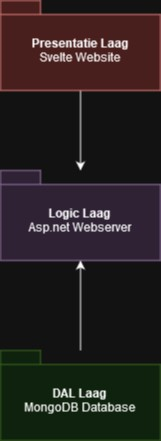
[Level 3 4](#_Toc151988679)

[Level 4 5](#_Toc151988680)

[Database design 6](#_Toc151988681)

[Design 7](#_Toc151988682)

# Layer Diagram



This diagram presents a three-layer architecture for a Web application. The top layer known as the Presentation Layer employs a Svelte framework for the AI scanners user interface. The middle layer known as the Logic Layer uses an ASP.NET Web server that is responsible for handling business logic. Finally, the bottom layer is the DAL also known as Data Access Layer, is where the MongoDB database is located, which manages data storage and organization with the use of Entity Framework.

# C4 model

## Level 1: Context Afbeelding met tekst, schermopname, visitekaartje, Lettertype Automatisch gegenereerde beschrijving

The C4 Level 1 shows the big picture of what the AI Scanner does and what roles the user and administrator have.

## Level 2 : Containers

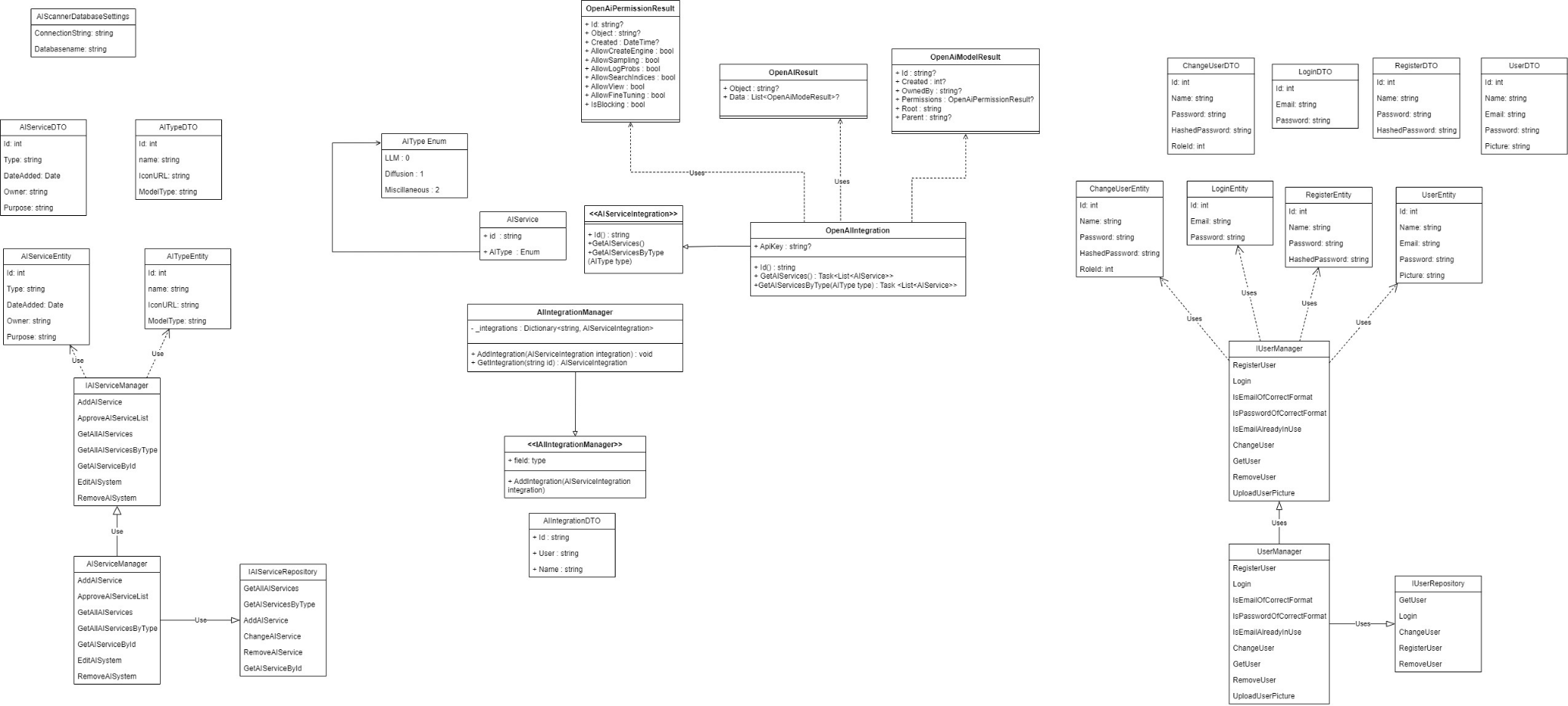
The C4 Level 2 diagram outlines the software architecture of an AI Scanner with two types of users: regular users and administrators. Both users access the system through a Web application built with SvelteKit, which serves as a login page. After authentication, users interact with a single-page application that displays AI services and integrations. This application communicates with an API application built with ASP.NET Core API with .NET 7.0 that provides the backend functionality for the AI Scanner. The API application communicates through API keys with AI services from other companies. The API application also communicates with the MongoDB Database that stores information about AI services and authentication.

## Level 3 : Components

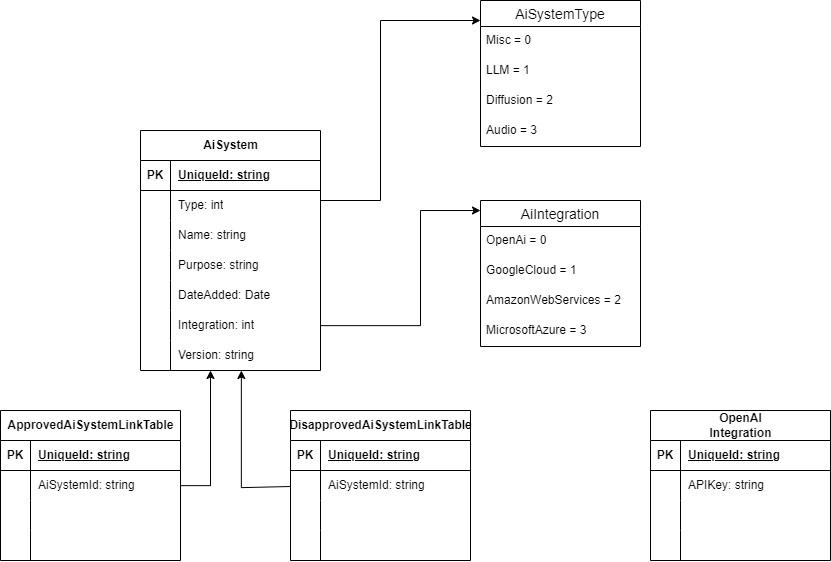


The C4 Level 3 model illustrates the workings of the AI Scanner. It outlines how the AI Scanner application interacts with an API Gateway that links to two components: the Account Controller, which manages user logins and the AI System Controller which registers AI systems. Both controllers communicate with a MongoDB database that stores data for AI systems and user accounts.

## Level 4 : Code



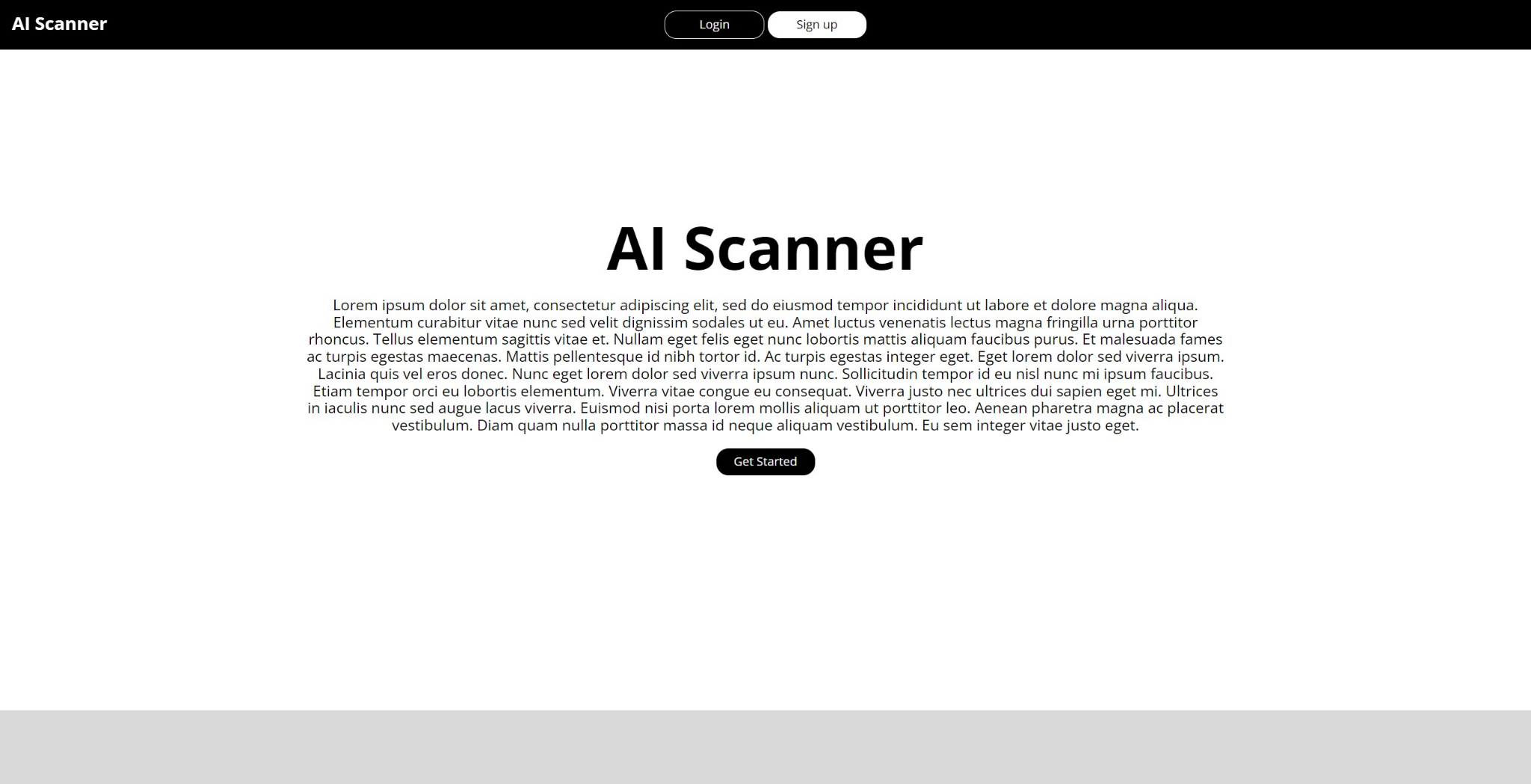
# Database design



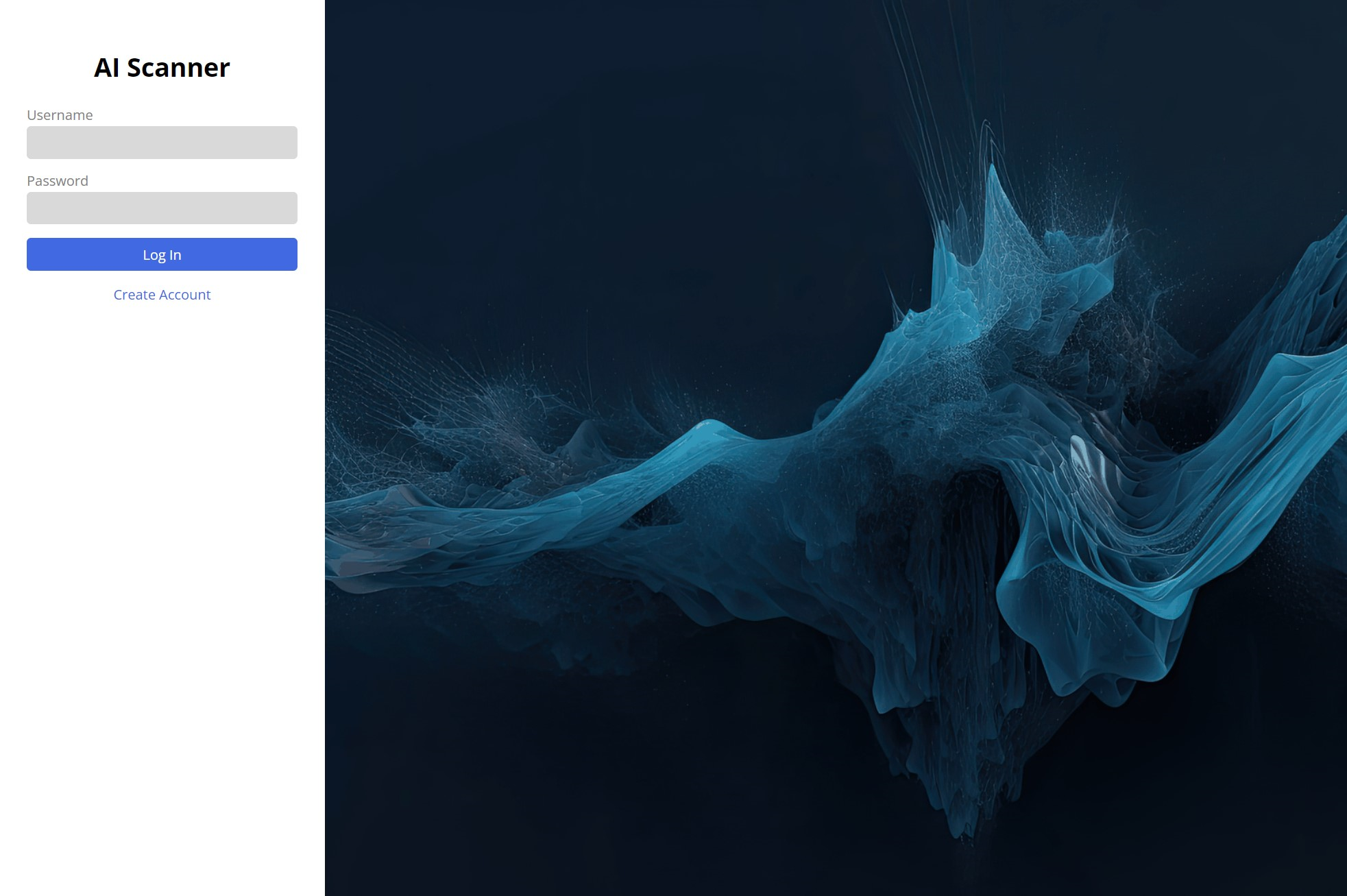
The entity relationship diagram revolves around the "AiSystem" table, which serves as the core for managing AI systems. "AiSystemType" and "AiIntegration" act as enums and define the set of allowed values for the Type and Integration attributes in the "AiSystem" table. There is functionality for approval and disapproval, represented by the "ApprovedAiSystemLinkTable" and "DisapprovedAiSystemLinkTable," allowing decisions to be made on whether or not to use AI systems. The schema also allows API keys for AI services to be stored in a separate table, such as the "OpenAI Integration" table. This structure facilitates not only the categorization and management of AI systems, but also the control of their approval and the secure handling of sensitive access data.

# Design

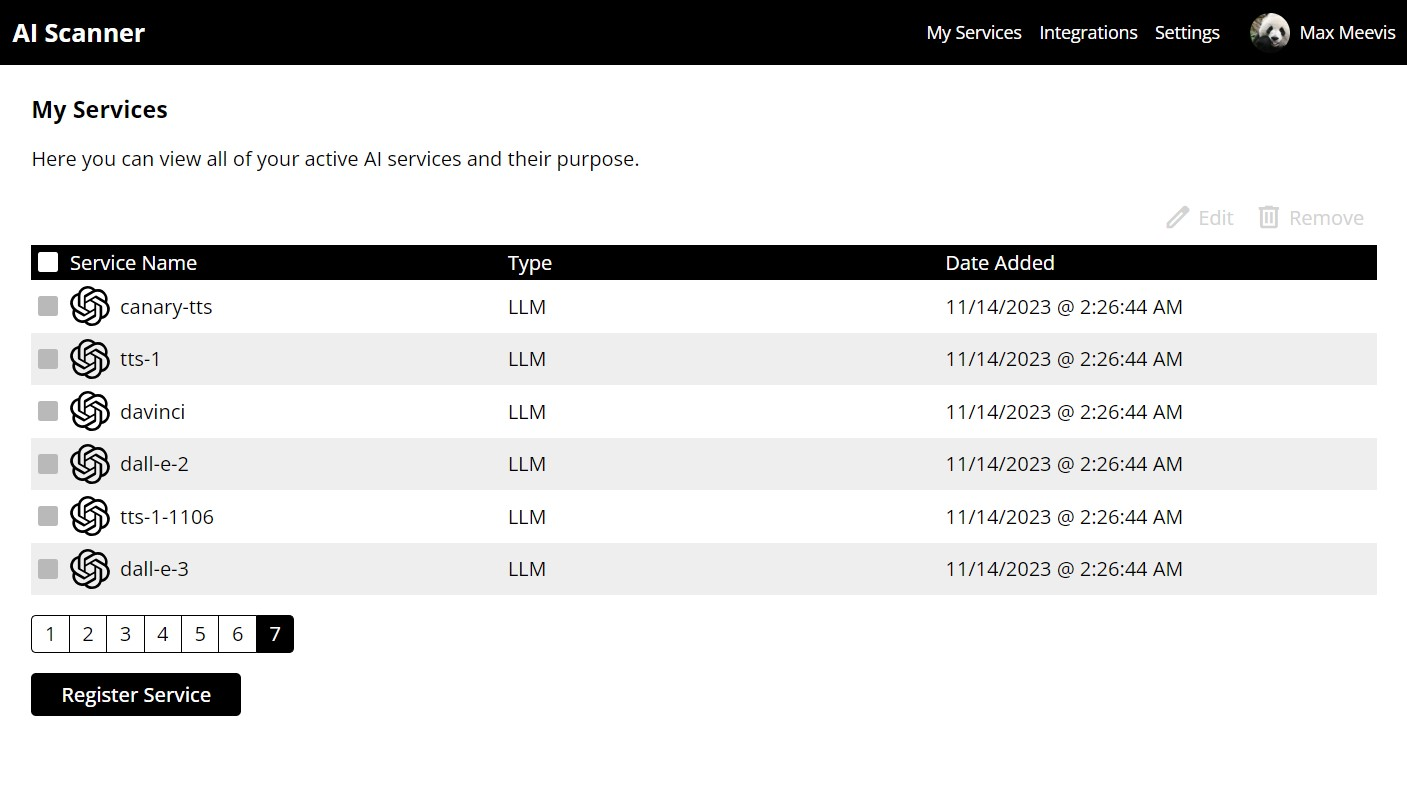
**Starting page:**



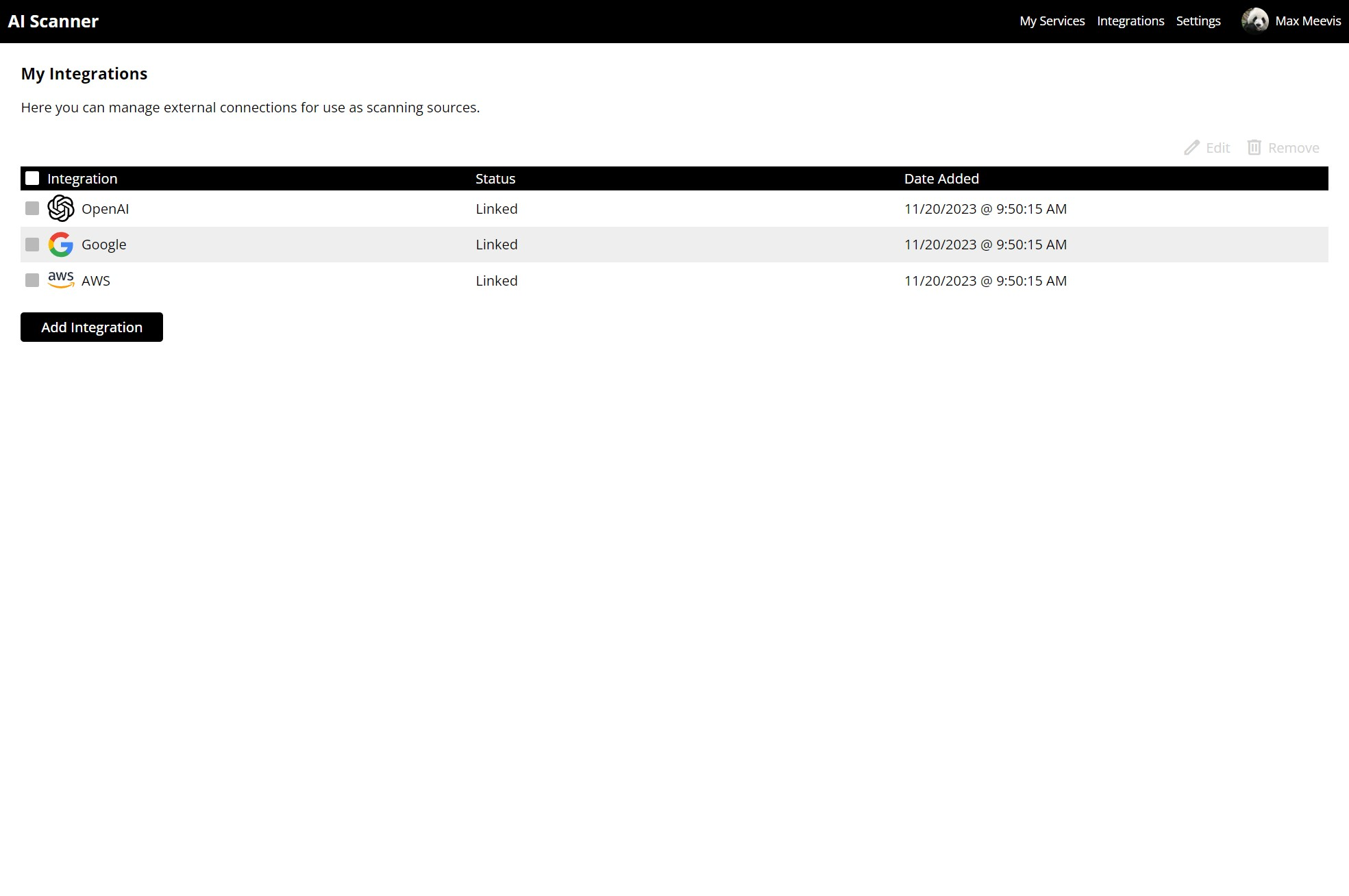
**Login page:**



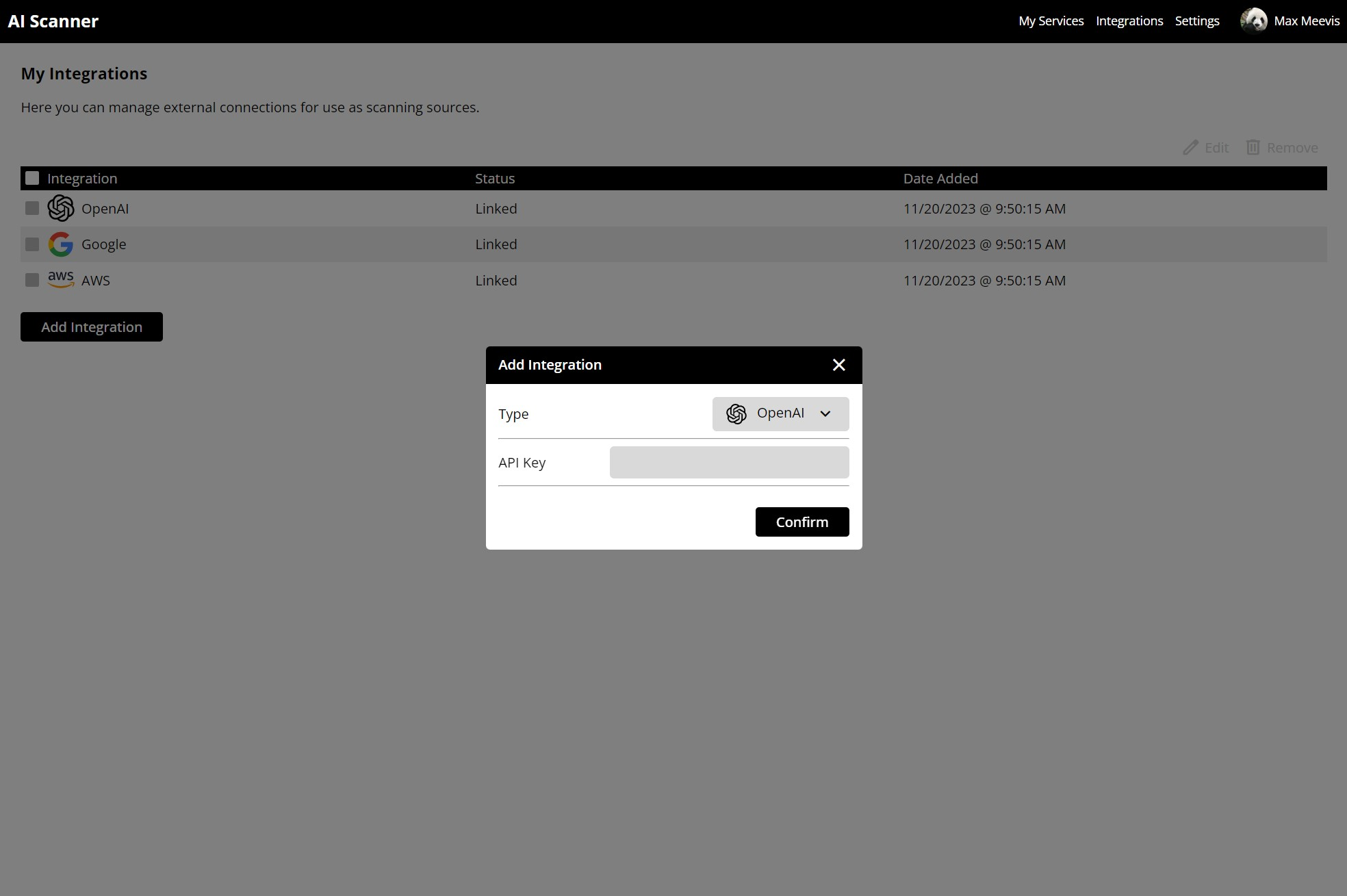
**My AI Services Dashboard:**



**My AI Integrations Dashboard:**



**My AI integrations: Add Integration pop-up window:**

  
**Profile page:**