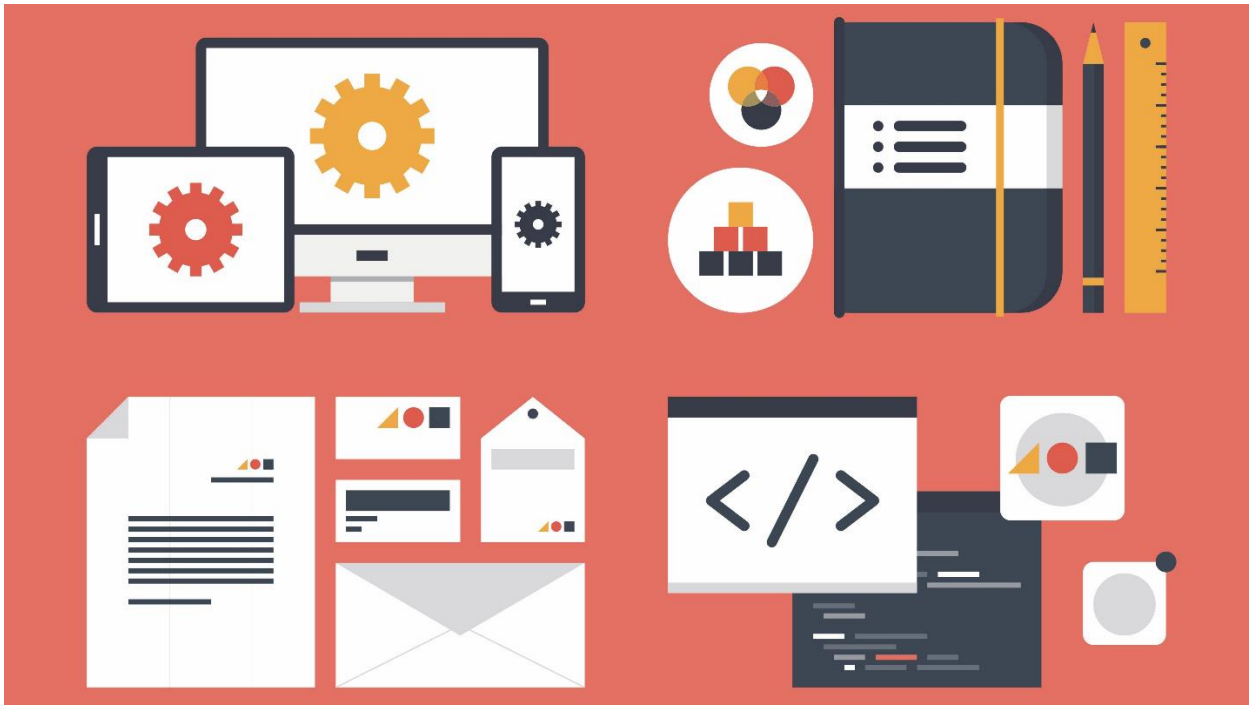
DESIGN DOCUMENT



Name: Tony Jiang

Date: 19 Sep 23

Project: Video call system

Versions	Date	Description
0.1	19 Sep 23	Initial document
0.2	6 Oct 23	Correct spelling errors. Explained the UML diagram. Make single quotes to double quotes.
0.3	12 Oct 23	Change the ERD section.
1.0	13 Oct 23	Initial documentation complete.

Contents

Introduction

This document covers all the technical aspects of this project, including its structure and the design decisions made throughout the development. It helps to further reflect on the design choices within this project. By doing so, it helps with planning how everything should be configured and address any potential development challenges. Additionally, it helps in conveying the intended design to other developers, ensuring a shared understanding and agreement on the design approach. Some information may refer to other documents where the information is fully detailed. New content or components will be added to this document throughout the project, and it is intended for technical developers.

The project's objective is to create two or three solutions for a video call system within the PRAS application.

Context

This is to give general information about the project. What's the project about and from whom. For context, please refer to the **Project Plan** document, on "*1.1 Context*".

Goals

This is to give information about what the project is trying to achieve and what the purpose of the project is. For the goals of this project, please refer to the **Project Plan** document, on "*1.2 Goal of the project*".

Scope

This is to define what needs to be delivered for the entire project, to ensure clarity for all stakeholders. The scope of the project can be found in the **Project Plan** document, on "*1.3 Scope and preconditions"*.

Functionality

This is to define the core functionality of the system, so that it's clear to everyone. To achieve a clear understanding of the functionality of the system, we use something called "user story". User story describes the functionality of the product, the expected behavior, and the required components. You can find the user stories in the **User Story** document.

System Architecture

The purpose of System Architecture is to describe the internal system's overall structure and establish an agreement on the desired design of the system. To make it easy to describe and communicate the system's architecture, we'll use a C4 architecture diagram. It's an architecture design that is easy to understand. The design approach is straightforward and helps us communicate how each part of the system should be set up, even to a non-technical person. It's like using Google Maps and you're trying to zoom in on a place in Aruba. The closer you zoom the more details you see. It's like that. The C4 architecture diagram helps us do the same for our software's structure.

The C4 architecture diagram has 4 level:

- Level 1: System Context (C1)
- Level 2: Containers (C2)
- Level 3: Component (C3)
- Level 4: Code (C4)

We will go to each level and describe what they do when we reach there. The tool that was used to create the C4 model is Visual Paradigm online. The free version.

Level 1: System Context (C1)

At this level you'll discover which intended user is going to use the system and what the system does. This provides an overview of the entire system. Additionally, it helps to describe what the system does to non-technical people.

Figure 1: C1 model

In the C1 model, you can see that employees can make video calls to retirees using the video call system. This provides an understanding of which users can use the system, what kind of system it is and the action they perform on the system.

Level 2: Containers (C2)

At this level, we delve deeper into the system's architecture, moving beyond the high-level system context of level 1. Here, we identify the high-level technical building blocks or "containers" that make up the system and understand the relationship between them. These containers represent the applications or databases used to build the entire system. It's important to note that the information presented from this level onwards is intended for the technical audience.

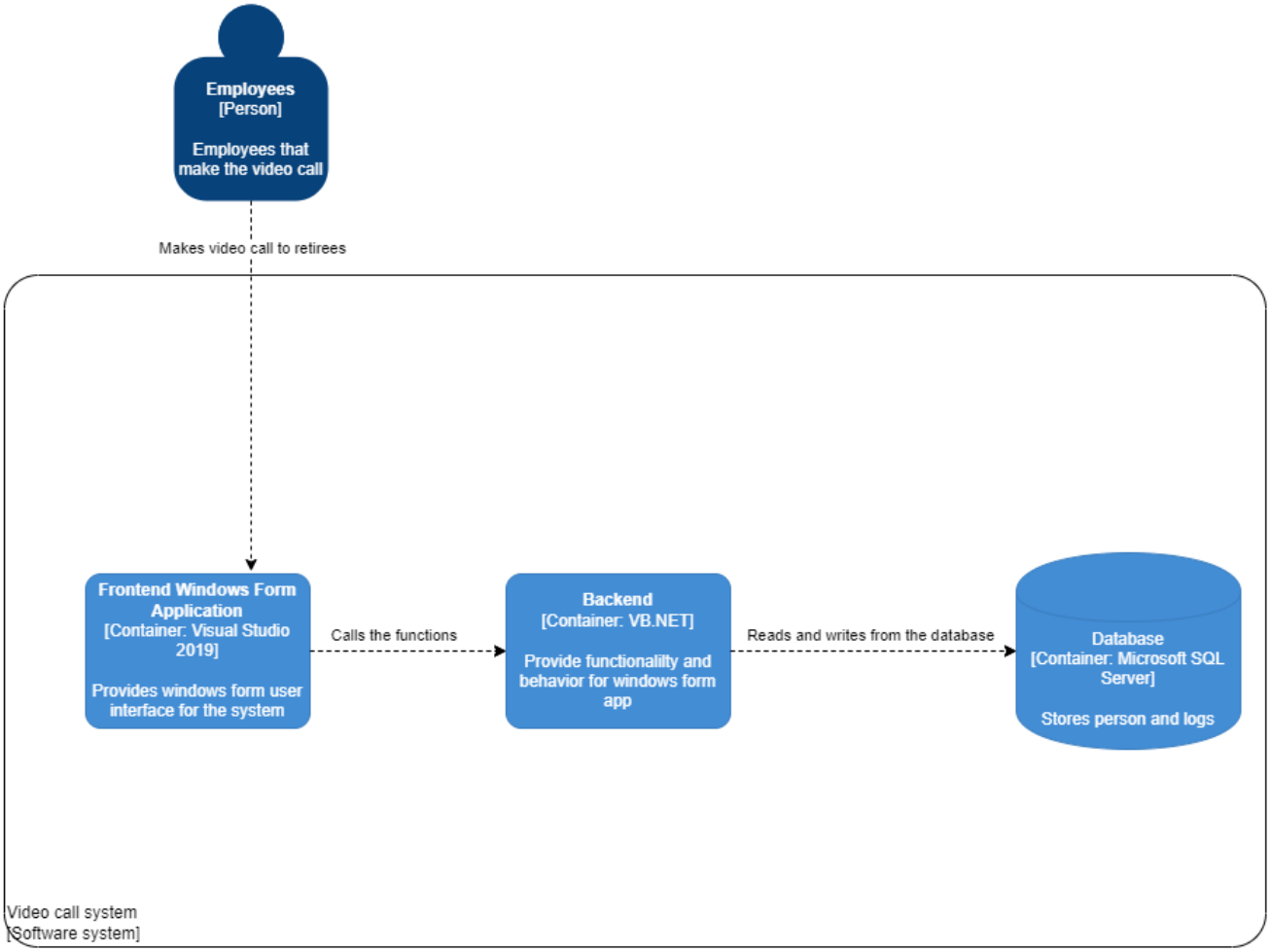


Figure 2: C2 model

It's pretty understandable how the system is put together from the C2 model. There are 3 containers that make up the system.

- **<u>Frontend Windows Form Application</u>**
    - This component serves as the user interface, where the users interact with the system.
    - It communicates with the backend to process user's actions.

      <u>Technology: Visual Studio 2019</u>
        - The PRAS application is developed in Visual Studio 2019
        - The PRAS application uses Windows Form App
        - It ensures seamless integration within the same IDE for the video call system.

- **Backend**
    - "All the functional code responsible for the system's operations, resides here.
    - It manages data flow between the front end and the database.

      Technology: Visual Basic .NET (VB.NET)
        - The backend is written in VB.NET language for the PRAS application.
        - Utilizing the same language for simpler integration into the PRAS application.

- **Database**
    - This container stores all the retirees' information and video call logs.
    - It provides data to the backend as needed.

      Technology: Microsoft SQL Server
        - The PRAS application relies on Microsoft SQL Server as its database system.
        - Utilizing the PRAS application's existing database simplifies the process of fetching and adding data and avoid the necessity of making a new database and populating it.

Level 3: Component (C3)

Level 3 represents the component (C3) of the container from level 2 of the system's architecture. These components are the building blocks that make up the containers of level 2 and they interact with each other. These components are categorized by the function they are assigned to do.

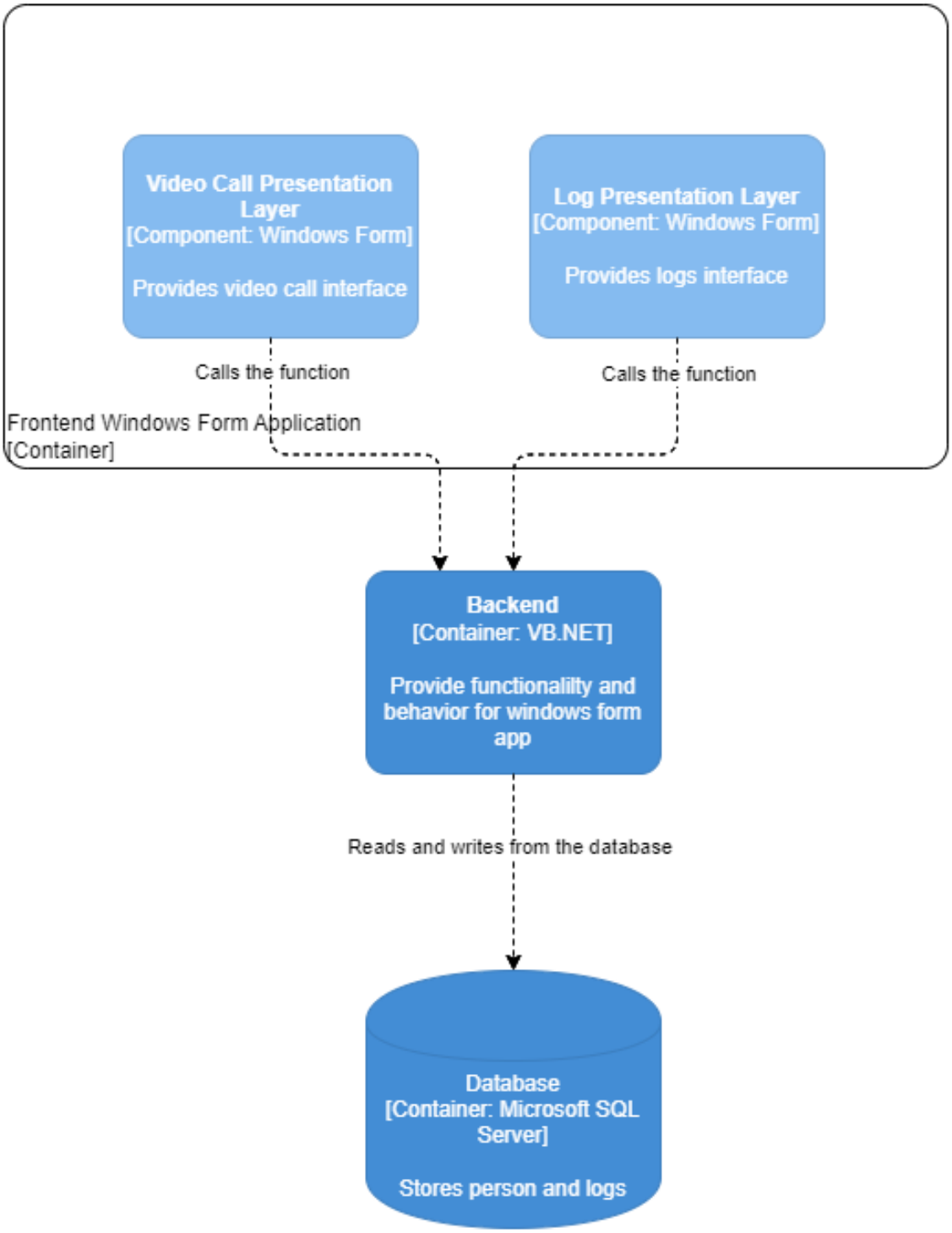


Figure 3: C3 model from Frontend Windows Form Application

This is the C3 model of the “Frontend Windows Form Application” container. It contains 2 components. The design idea here is to sperate the user interfaces with destinct functions.

- Video Call Presentation Layer: Here lies all the video call user interfaces
- Log Presentation Layer: This contains all the logs related to the user interfaces.

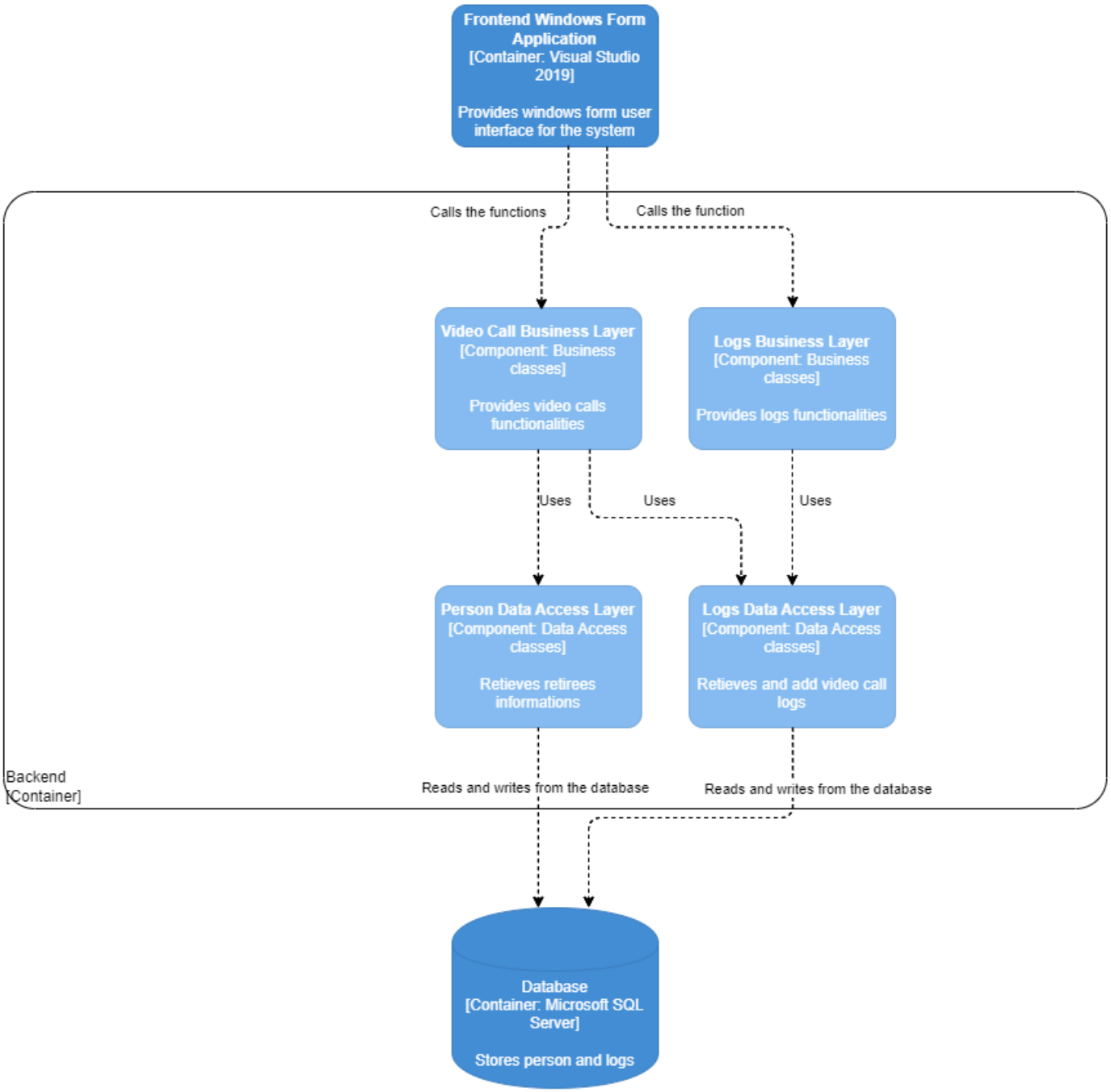


Figure 4: C3 model from backend container

This diagram shows the structure of the Backend container, which consist of 4 components that do their own task. This separation of tasks simplifies the understanding of the design and for future additional components.

- Video Call Business Layer
  - Contains all the video call functions.
  - It uses the “Person Data Access Layer” to retrieve retiree data.
  - It uses the “Logs Data Access Layer” to add logs to the database.

- Logs Business Layer
  - Contains all the logs functions.
  - It uses the “Log Data Access Layer” to retrieve and add log data.

- Person Data Access Layer
  - Manages data flows from the database to specific video call functions.
- Logs Data Access Layer
  - Manages data flows from the database to specific log functions.

Level 4: Code (C4)

This level includes UML (Unified Modeling Language) diagrams. UML diagrams are a type of visual tool used in software engineering to illustrate the structure and relationships of classes within a system or software application. They provide a high-level overview of the system by displaying the attributes and methods of each class within it. UML diagrams are valuable for planning and conveying how a system is built, and they are easily understandable, even for newcomers.

In our documentation, each component will include a UML diagram. This level is typically considered optional and is primarily used to illustrate complex class structures and explain their functionality. It will be used to depict the structure of the video call system in the backend and provide insights into the design reasons.

Please note that this UML diagram will be continually maintained throughout the project, ensuring that it accurately reflects the system's structure. We will be using Lucidchart as the tool for creating and updating these UML diagrams.

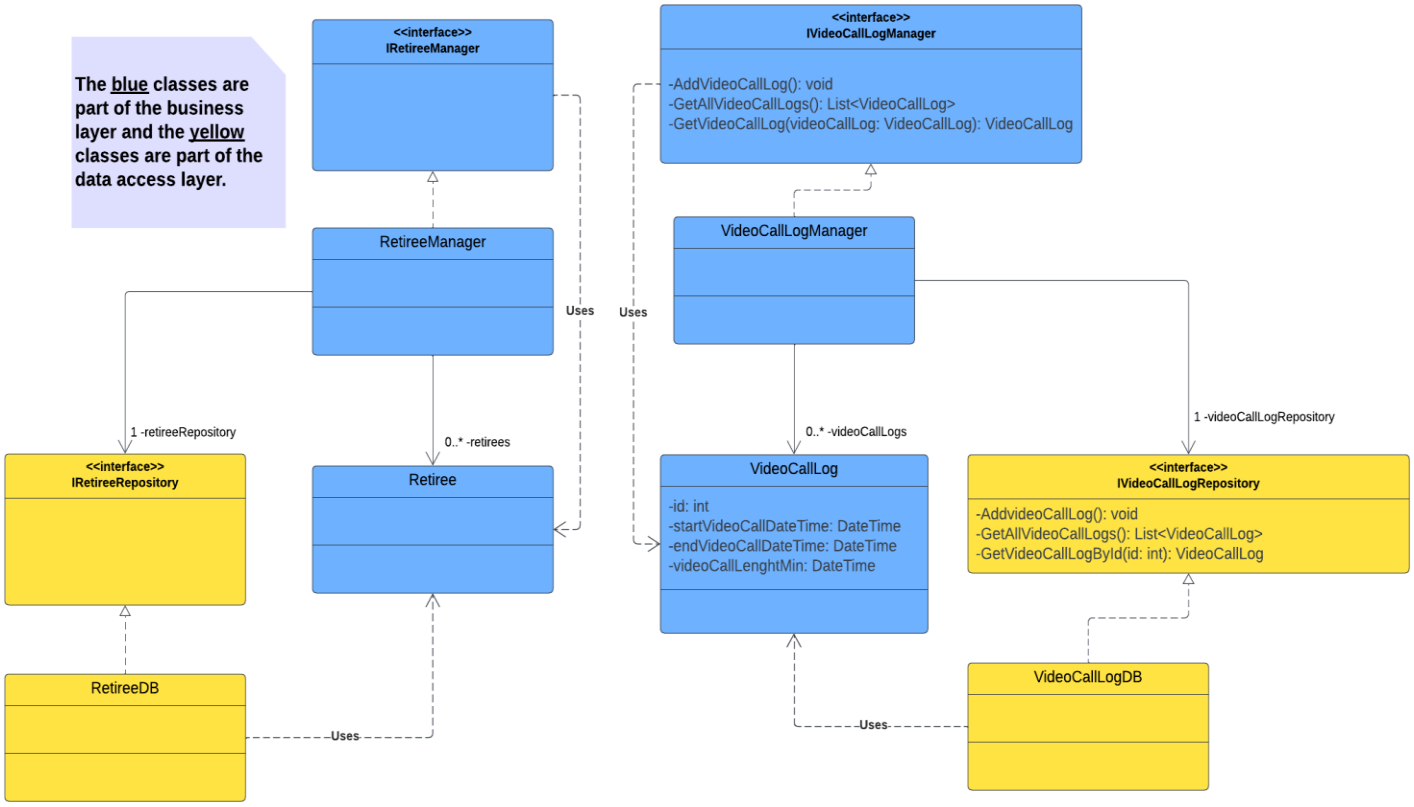


Figure 5: UML diagram

The diagram is structured to follow the SOLID principles, which helps in maintaining, extending, and understanding the UML design. This approach helps in building an adaptive, effective, and agile software application, primarily in object-oriented software development. Additionally, it simplifies the process of extending, modifying, testing, and refactoring the code.

SOLID stands for:

- **S**ingle Responsibility Principle

It states a class should only have one reason to change, meaning it should have a single job or a single responsibility. The UML diagram follows this principle. For example, the “VideoCallLog” class is responsible for the core data and actions related to individual video call logs, while the “VideoCallLogManager” class handles higher-level logic and operations for managing collections of video call logs, such as “AddVideoCallLog” and “RemoveVideoCallLog” methods.

- **O**pen-Closed Principle

It states that classes should be open for extension but closed for modification. This means that they should be open to extending their functionalities but closed to making changes directly to the class's code. It is implemented in this UML diagram. For example, the "IVideoCallLogManager" Interface can add new functionalities and can be implemented by other classes without altering the existing functionalities in the interface or the "VideoCallLogManager" class.

- **L**iskov Substitution Principle

It states that objects of a derived class should be able to replace objects of the base class without affecting the correctness of the program. In other words, the sub-class that extends from the base class should inherit the functionalities and properties from the base-class and should be able to replace methods without causing any problems for the base-class. This is not implemented in the UML diagram because currently there are no classes that inherit from a base class.

- **I**nterface Segregation Principle

It states that a client should never be forced to implement methods of an interface that it doesn't use. In other words, an interface should be specific to the needs of the implementing class. In the UML diagram, this principle is implemented with the "IVideoCallLogRepository" interface, which is designed specifically for the "VideoCallLogDB" class. No other classes implement this interface.

- **D**ependency Inversion Principle

It states that High-level modules should not depend on low-level modules. Both should depend on abstractions, such as interfaces or abstract classes. In other words, classes should depend on interfaces or abstract classes instead of concrete classes and functions. It encourages a more flexible and maintainable software design by reducing tight coupling between components and allowing for easier substitution of implementations. This is implemented in the UML diagram, as you can see that the "VideoCallLogManager" class implements from the "IVideoCallLogManager" interface. You can create another class to implement this interface.

User Interface Design

The purpose of this is to present the user interface to all the stakeholders and reach an agreement on the product's visual design. For illustrating and conveying the idea, we'll make wireframes and user flowcharts.

Wireframe

This is to illustrate the user interface design, we made use of wireframes, which are mockup designs of an application or website. There are websites or applications that allow you to easily create wireframes for free. You can create, export, demonstrate, and get feedback on it instantly. With the wireframe you can also provide detailed descriptions for each component on the wireframe. The wireframes can be found in the **Wireframe document**. Tool used for creating the wireframe: Figma.

User Flowchart

The user flowchart is a flowchart that illustrates the interaction between a user and the system. It provides visual representation of how they interact with each other. It serves as a useful tool for all stakeholders, including non-technical ones, to understand and communicate how these interactions should function. The user flowcharts can be found in the "User Flowchart" document, which was created using the draw.io tool.

Database design

This process helps us design, identify, and manage the database. It involves designing the overall table structure, identifying the types of tables required for the database, establishing the relationships between them, and maintaining and improving the database's design and data structure. The database design we are using is based on the Entity Relation Diagram (ERD). It gives a better understanding of how the database is designed. Some entities that are in the diagram are predefined in the company's database, and we will utilize these entities to retrieve data. Our focus is to design the database for the video call system in the PRAS app. The tool for creating the ERD is an inhouse company used tool: Aqua Data studio.

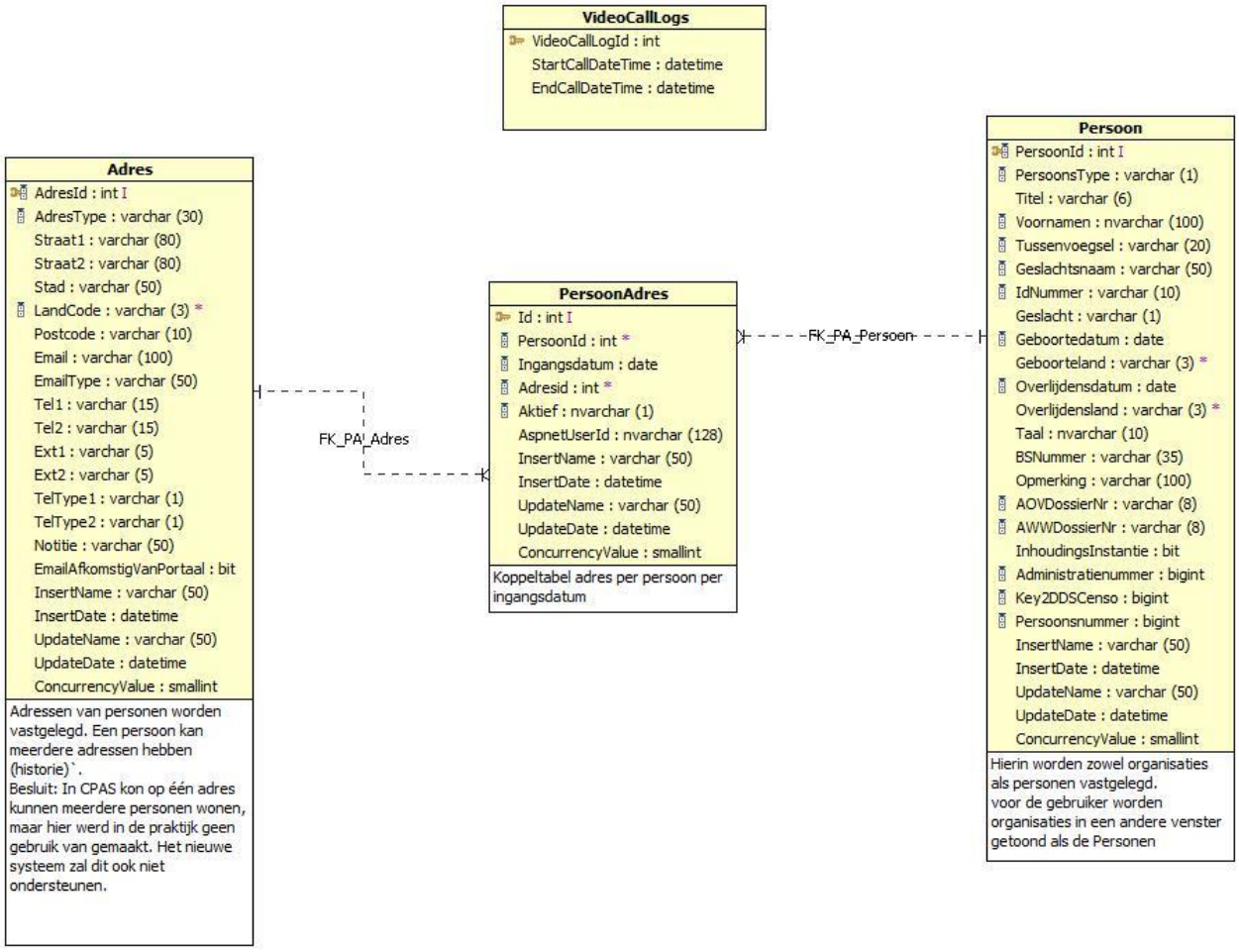


Figure 6: ERD

Some entities that I'm going to use is already define in the companies database and they are written in Dutch. The "Persoon (Person)" , "PersoonAdres (PersonAddress)" and "Adres (Address)" are entities from the company's database. They are used to get retirees' information. The "VideoCallLogs" entity is used to store the data of the video call logs.

Testing strategies

This is to plan out how I'm going to test the video call system, what type of strategy I'm going to use for testing, and the reasons for the tests. For these tests, you would need to follow step-by-step instructions on how to test the system. The summarized version of testing strategies is in the project plan document in section 4.1," Testing strategies".

Test approach

Test type	Explanation	When should you start	Test written by	Tested by
Unit test	This is to test the functionality of the system's components and ensure that everything is working as expected, even after a new line of code is added. It serves to prevent any code error.	This should be written at the start of the initial documentation phase. To get a clear idea of the possible problem that may occur to the application. After that, it can be written whenever and it can be tested at the end of each sprint.	Tony	Tony
User acceptance test	To test the expected user outcomes and prevent any unexpected results to show on the system.	The test cases need to be written before testing. After each function is finished on the user interface, we conduct the test at the end of each sprint.	Tony	Edwin

Unit test

Here, we will be defining all the unit tests that need to be written and tested on the system. Please note that new unit tests may be added, modified, or deleted as the project progresses.

Method for getting phone number	Status (done or not yet)
ShouldGetPersoonPhoneNumbersByTelType()	Not yet
ShouldGetPersoonPhoneNumbersByStartingDate()	Not yet
ShouldGiveAnExceptionWhenThereIsNoPhoneNumber()	Not yet

Method for calling the video call	Status (done or not yet)
ShouldInitializeVideoCall()	Not yet

Method for ending video call	Status (done or not yet)
ShouldEndVideoCall()	Not yet

Method for getting email	Status (done or not yet)
ShouldBeAbleToGetPersoonEmail()	Not yet
ShouldGiveAnExceptionWhenThereIsNoEmail()	Not yet

Method for adding date and time of the log	Status (done or not yet)
ShouldBeAbleToAddDateAndTimeOfLog()	Not yet

Method for getting the date and time from the video call	Status (done or not yet)
ShouldBeAbleToGetTheDateAndTimeOfVideoCall()	Not yet

Method for microphone	Status (done or not yet)
ShouldBeAbleToMuteMic()	Not yet
ShouldGiveExceptionWhenMicNotFound()	Not yet

Method for camera	Status (done or not yet)
ShouldBeAbleToTurnCameraOff()	Not yet
ShouldGiveExceptionWhenCameraNotFound()	Not yet

User acceptance test

Here, reside all the user acceptance tests and will be tested once each function is complete at the end of each sprint. The written user acceptance tests will be stored here, and the test

reports for the user acceptance tests will be located in the “Test Report” document. Please note that new user acceptance tests will be added, modified, or deleted as the project progresses.

Created date: 27 Sep 23

ID	User story	Description	Expected result
1	US 1	This is to see if a from appears.	A form should appear
2	US 2	This to see live feed of the video call.	I can see the live video call.
3	US 2	This is to see if you can hear any sounds from the live feed.	I can hear the sound from the video call.
4	US 2	This is to see if your camera is working.	I can see my camera on a small screen in the video call.
5	US 3	This is to see if the logs of the video call are written.	I can see the log of the started and ended video call by date and time.
6	US 4	This is to see if you can end the video call on the end video call icon.	The video call will end, and the form will close/ disappear.
7	US 4	This is to see if closing the form would end the video call.	The video call will end, and the form will close/ disappear.