Linnæus University
Sweden

Assignment report

1DV700 - Computer Security
Assignment 3
Software design document

Author:
Hussam AlKHAFAJI - ha223cz
Paolo Molinaro - pm222py
Rutger Nieuwenhuis - rn222np
Antonio Wikstrand - aw223pi
Term: HT20
Course code: 1DV700

Contents

1 Introduction

This chapter should provide an overview of the entire document and a description of the scope of the system and its intended usage.

1.1 Purpose

This section is intended to describe the purpose of the software design document and the target readership that it is meant for.

This design document was produced upon the request of Loco news as a guideline or a checklist for Loco news as well as the developers working on the application that is described in this document. This document shall give detailed explanations starting from the General overview of the whole system, going through System design and finishing off with some use cases that might be helpful for the developers and Loco news alike.

The document will have a special focus on security and might sometimes mention security concerns that may not be directly linked to the application but their consequences affect the security of the application on the long run.

1.2 Scope

This section should present the products that are going to be produced from this document. It should also describe the benefits, objectives and goals that the system would provide.

The product introduced will fulfil the requirements mentioned by Loco News introducing a design tailored for their needs with major focus on security and efficiency. The objective of the design is to build a strongly secure and simple cloud-based layered architecture combined with a modular object-oriented structure, introducing various security-wise and efficiency-wise improvements for both on-site and off-site employees' usage. Follows an overview of the objectives, with relative benefits and goals:

The main objective is the migration of the database to the cloud, to lift the security responsibility from the essential IT team employed and to resolve the unsuitable location of the servers, guaranteeing that the safety of their data is commissioned to hosting company. The existing physical database will be kept for cache of frequently accessed files, available only on-site, to improve performance of the cloud storage.

The program will be built with a layered and object-oriented structure, while keeping a modular approach, that will allow the developers to identify the most critical operations as well as allow an easy swap and update of modules.

An open design software that will be focused on access control and an ease of use design that will effortlessly implemented in the already existing environment, while introducing proper security mechanisms needed like: digital certificates and signatures, authentication exchange and cryptography, privilege and access control; given the sensitivity of the information treated by Loco News.

2 System overview

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

2.1 General overview

Briefly introduce the system context and the basic design approach or organization. Provide a brief overview of the system and software architectures and the design goals.

2.1.1 Context

This system is supposed to meet the requirements mentioned earlier which consist of a program that fulfils Loco news needs for fetching, updating and storing information as a way to increase productivity.

2.1.2 Design approach and organization

The developers are advised to follow the following approaches to maximize security and minimize the chance of a failure or a fault.

When it comes to the design the developers should consider the following rules:

- **Least privilege**: The program should be multi-user (e.g. admin, editor, reporter, photographer) and should give each user the least amount of privileges possible to achieve their job. We will discuss how to implement this feature securely in the software architecture section. Privileges must be periodically updated so that the program provides a safe and a fast way to update privileges.
- **Economise the number of mechanisms**: Any component of the security mechanisms of the program must be as small as possible to allow it to be analysed and verified.
- **Open design**: The program must not rely on security through obscurity! The program should only depend on the secrecy of certain delicate items (e.g. passwords, keys, etc.) even if the source code of the program is open to public.
- **Complete mediation**: Every access attempt must be checked by the program, whether it's a direct access attempt or an indirect access. The program must not rely on anything other than its security components to achieve mediation. The program must also implement ways to mitigate incomplete mediation such as signing access attempts to make sure they're not tampered with while the system is authenticating.
- **Permission based**: The program should be permission based and the default should be the denial of access. The program should also re-authenticate when a critical operation is called upon. Critical operations are commands

that would lead to catastrophic results such as any interactions with the database.

- **Separation of privilege**: Mediation should depend on at least two methods of authentication. The method of authentication consists of:
  - What the user knows (e.g. password).
  - What the user has (e.g. token).
  - What the user is (e.g. bio-metrics).

  Extremely critical operations should depend on more than two conditions.
- **Least common mechanism**: The system should provide logical separation of its components and modules to minimize the risk and consequential damage of a failure. The advised modules will be described in the software architecture part.
- **Ease of use**: The system should be easy to use, especially the security parts, because if they're cumbersome, it will lead to users avoiding them which will condition users to misuse the system. This will create failures even if the modules are outstanding.
- **Avoid penetrate and patch**: The developers are better off by starting with developing the security modules first and then all the other modules around it. The approach to stick security at last like an add-on is known as bad practice and guarantees failures.

2.1.3 Overview of the system and software architecture

- **Architectural patterns**
  - **Global pattern: Client-Server**
    The program is a client server program. The client side is the actual program and the database is the server side. Examining the situation at Loco news, we have decided that the best course of action here is to employ a cloud-based server from a reputable enterprise to host all the data from the company. Our reasoning is that because of the low number of capable IT employees, the amount of sensitive data in the server and the unsuitable location of the server, it will be hard for Loco news to guarantee the safety of their own data. All the security related to the server side should be handled by the hosting company. Loco news should consider taking insurance to mitigate all the risks by transferring them to others. The actual server that exists inside the Loco news building should first be separated by a fireproof wall and all the locks should be changed. That server will be only used for temporary storage of information to improve performance using the cloud storage.

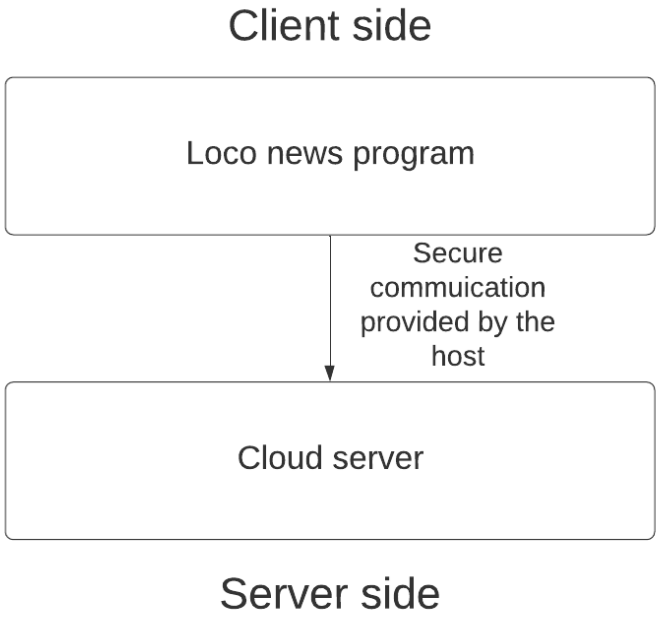

Figure 1. Client-Server pattern

- **Other design structures**
  - **Layered structure**
    The program should be modelled in layers where each layer has a logical separation from the layer underneath it. However, every layer depends on the layer below it. The most important layer is the bottom one and the least important layer is the uppermost one. This design will allow the developers to identify the most critical operations; The operations that belong to important layers where failure will compromise most of the other layers.

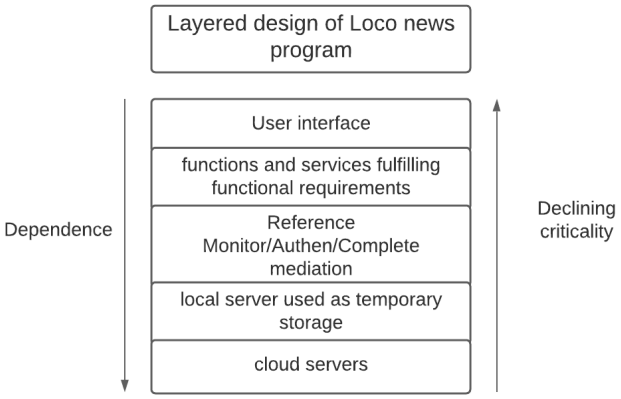

Figure 2. Layered design of Loco news program.

The layers in the figure are:

* **User interface**: Containing all the modules that interact directly with the user such as the Graphical user interface.
* **Application and Business layers**: The three next layers are all part of the application and business layers. Each one of those layers should contribute to the completion of the user requests while keeping their work as separate as possible. Inside these layers there is a place that consist of a security layer that must take care of every security function from access control to auditing (reference monitor).
* **Persistent storage layer**: This layer consists of functions that interact with the cloud server we discussed earlier as well as all the security requirements those functions entail.

– **Object-oriented structure**: The program should follow an object-oriented design. This entails that the program is structured into modules (inside the previously discussed layers). Each module should have one single functionality and should have low dependency on other modules. The modules should be designed with high cohesion and low coupling in mind to guarantee minimized risk on all layers. Security modules should especially use encapsulation to hide implementation information. This approach will allow an easy swap and update of modules in the future which will lead to easier maintenance. Security modules should be developed using the knowledge of security experts as well as cryptography experts. The developers should use known and well-established cryptography methods instead of implementing their own. They should follow the cryptography expert's indications on how to store passwords and other critical information using encryption. Those passwords and other information used to authentication should be stored in Loco news server in accordance with the methods the experts suggest.

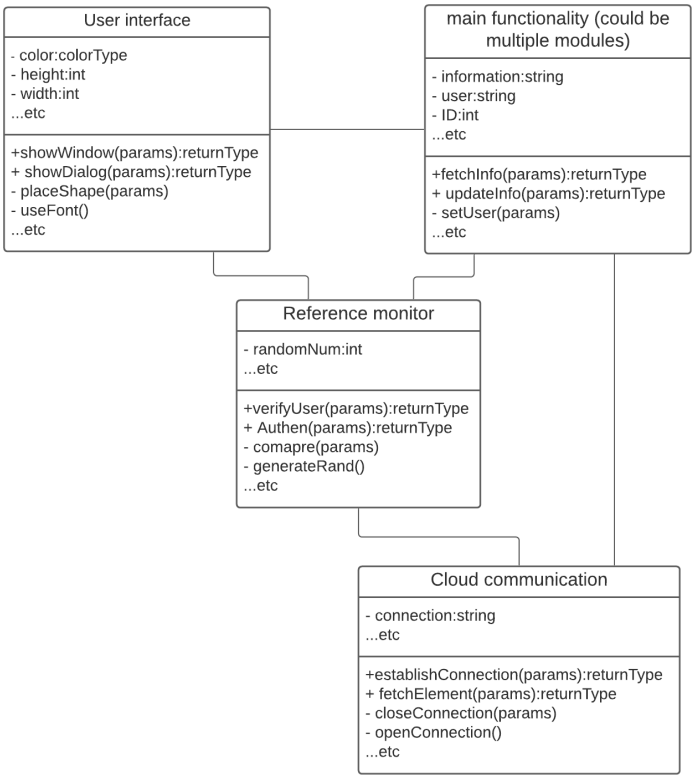

Figure 3. Global modules of the program

This figure shows the global modules of the program. Each global module is like a package that must be broken down to several modules where each module is only responsible of one and only one functionality.

2.2 Assumptions

Describe any assumptions or dependencies regarding the system, software and its use.

The program and developers should assume certain ideas such as:

- The cloud server is to be trusted as the host guarantees top notch security, given that Loco news will contract a reputable enterprise that have experience with cloud-based services. (IBM, Google, Microsoft, Nvidia, etc.). The host will guarantee integrity, confidentiality and availability of data. This cloud server will be used to store data as well as back-ups.
- The program should assume distrust of everything starting from its modules to the users. This means the program should sanitize input between modules and sanitize input from users as well.
- Not a single security module is allocated to certain developers. There should always be several teams working on a given security module. This lower the risk of intentional tamper from one developer or a group of developers.

2.3 Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's hardware, software and/or communications, and describe the associated impact.

With the design approach, organization, and the software architecture described in the previous chapter, new constraints and limitation are introduced.

Limitation related to the design:

The company will have a secure and proper access control, compared to the situation discussed in the interview, so every user will have certain (as limited as possible) privileges and it will take time for each user to get used to it, including having to ask for new permissions if needed, and paying more attention to the credentials as it will be stated in the new policy.

Limitation related to the server-side:

The network connection dependency is one of the biggest constraints of the architecture chosen, meaning that in order to obtain the benefits of cloud computing, Loco News must always rely on an internet connection. Thus, it will be limited on the bandwidth provided, even if nowadays, providers supply unlimited bandwidth.

If the company loses the network connection for any reason, it may experience downtime; however, we expect that the hosting service will provide a continuity plan and an SLA close to 100% up-time.

Another limitation comes from the loss of control, meaning that the company is entrusting the integrity and protection, both physically and online, of the data to the hosting service provider. If a technical issue occurs the company has to rely on the customer service of the provider, even if IT staff is present in-house.

The final limitation, related to the database, resides in the initial lock-in due to the first migration, that will make impossible to log into it during the whole data transfer.

Limitation related to the client-side:

- **Layered architecture:**
  The main limitations of a layered architecture are: the lack of inbuilt scalability, therefore, it must be properly organized since the creation, and the correlation of each layer from the one below.
- **Object-oriented architecture:**
  The main limitation of the object-oriented architecture is that it is not popular, therefore, it will be hard to find a skilled team that can handle problems. It is important also to consider that, in some situations, the high complexity of the program can affect the performance.

2.4 Risks

Describe any risks associated with the system design and proposed mitigation strategies.

This design entails some risks such as:

- Integrity, confidentiality or availability failures from the cloud-based server.
  Mitigation strategy: Assurance and guarantees from the host enterprise to transfer the possible damages to them instead of Loco news.
- Local server at Loco news might be breached.
  Mitigation strategy: The server should not be directly accessible from the web; it should only be connected to the LAN. The admin is the one responsible for the safety of this server. The developers must consult cryptography experts to ensure the confidentiality of sensitive information in the server. Lastly, The IT team should use part of the server to create a honeypot that contains false information and fake but believable news.
- Misuse of the program from a trusted user.
  Mitigation strategy: By following the ISO IEC 27002 or 27001, Loco news should do screening all its employees. There should be several security logs that are kept safe separately from the known servers. Loco news should provide security training to all its employees. The program itself should not accept connection to public Wi-Fi networks. It should only connect to the company's LAN as well as private connections using USB ports that should be distributed to the employees that work from afar.

3 System design

This chapter should present to the software developers the mechanisms that should be implemented.

3.1 Software design

Describe all software that is needed to support the system, the hardware component for which each software component is targeted, and specify the physical location of all software systems.

The system contains three major components: A local database, a back-end application and a front-end application. All three components, their hardware components and their physical location are described below. In addition to that, some recommendations on programming languages and other development tools will also be given for all components. These are not the only valid suggestions, as there are many other possibilities that will work as well. They are intended to give an idea of development skills required to make the software component in question. Furthermore, the components will also be linked back to the layered design in section 2.1.3.

A cloud service is also required in the software system, as described in section 2. This cloud service is not included in this list, as the cloud service will be given by a trusted cloud service provider, and therefore does not need to be programmed. However, the cloud service is referred to when necessary in the list.

- Local database
  Hardware component: The server
  Location: The server room

  The local database is a critical component that will store all data locally. This can include employee information, new articles, etc. The database will only be accessed by employees working on-site. Employees working on distance will work directly to the cloud. This might take longer for the employees on distance, but it greatly adds to the security of the system, as this means that the database does not need to be exposed to the world.

  In the layered design of section 2.1.3, the local database is part of the application and business layer as the temporary storage before the cloud server. The local database will function as the temporary storage, with the advantage that data fetched from the cloud server can be cached and accessed quicker by the employees on-site. In addition to that, data uploaded by the employees will go to this database, which will upload it to the cloud. This will also save the employees time.

  An industry standard for database technology is the Structured Query Language (SQL). An advantage of SQL is that given its popularity, there are already many database software out there. Some of these are open source,

such as Oracle's MySQL [1]. Using this software does not require the organization to buy a license, or to hire anyone to program database software on its own. Furthermore, as MySQL is widely used, it is often examined and therefore, much more secure than programming your own database software.

To set up and maintain the SQL database, a skilled database administrator is required. The MySQL server can both be run on Windows and Unix. However, Unix is strongly recommended, as it is open source, and therefore does not require the organization to buy a license. Furthermore, Unix strongly supports server applications and is updated very frequently, making it a secure option to run the server on. In addition to that, Windows requires more system resources that Unix [2].

- Back-end application
  Hardware component: The server
  Location: The server room

  For computer systems to communicate with the database, a back-end application needs to be developed. This back-end application will function as a bridge between the SQL database and the user's requests to the database. This application needs to sanitize the user inputs, check permissions and return the requested data to the user.

  Using the layered design of section 2.1.3, the back-end application can be divided in three parts:

  - **Functions and services fulfilling functional requirements**
    First of all, the back-end application needs to be accessible to some way by employees. This will go via the front-end application, through the local network. The requests made by the employees via the network must be handled properly by the back-end application, and responses must be sent back.
  - **Reference monitor/Authentication/Complete mediation**
    The requests made by the employees must also be verified for security. The back-end application must authenticate the employee, and function as a reference monitor to see if the employee is allowed to access the requested resource or make the requested operation (complete mediation).
  - **Communication with the cloud server**
    Lastly, the back-end application needs to make sure that the data on the local storage is backed up to the cloud server on regular intervals. Furthermore, if the data requested by the employee is not present on the local database but on the cloud server, the back-end application needs to fetch this data from the cloud server and cache it in the local database if necessary. (Granted that the employee is allowed to access this information.)

Many cloud services can be accessed via an application programming interface (API), for example Google Drive [3]. In addition to that, SQL is a widely used industry standard. This leaves many possible programming languages suitable for this application. Two example languages are Java and PHP. Both languages have supporting packages to work with SQL [4], [5]. Furthermore, PHP is completely open source and Java is largely open source, allowing the organization to make an application without buying licenses [6], [7]. For Java, a Java virtual machine (JVM) is required to function as both a compiler to bytecode and interpreter for this bytecode. For example, OpenJDK as an efficient and open source option [8]. PHP will require an open source application such as Apache to run [9].

- Front-end application
  Hardware component: The employee's computers
  Location: Anywhere the employee takes their computer

  The employees need to access the database, both to upload their data and retrieve required data. This has to be handled by a front-end application. In the layered design of section 2.1.3, the front-end application is the user interface.

  Employees working locally will access the database on-site. Employees working on a distance will work directly to the cloud. Although this takes longer, this way the database does not need to be exposed to the world. The front-end application should detect if the employee is on the local network. If this is the case, it should communicate with the back-end application to interact with the database. Otherwise, the application should communicate with the cloud server.

  There are different environments used by the employees. Some employees use Windows, others use macOS or Unix. It is recommended that the organization switches the employee's computers to only using one environment, so that the front-end application will be easier to implement. The recommended operating system for the employees is Windows, as it is very user friendly. The organization will need to buy licenses for Windows on the employees' computers.

  Furthermore, as this application is used by employees directly, a graphical user interface (GUI) is required. This makes for an easy user experience for the employees.

  This still leaves a lot of options regarding programming languages. The Google Drive cloud service has an application programming interface (API) that can be accessed by programming languages for example [3]. In addition to that, many languages allow for GUI design. Java would be a suitable candidate, as it can communicate with an API, is cross-platform and allows for GUI design [10]. Another suitable candidate would be JavaScript with an appropriate framework, which also checks all three requirements [11].

To conclude, the front-end application communicates with the cloud server for off-site employees, and with the local back-end application for on-site employees. The back-end application then verifies the employee's request, and communicates with the local database if the request is permitted. If the required data is not present on the local database, the back-end application will communicate with the cloud server. Furthermore, the back-end application is also responsible for regular backups of the local database. The following flow-chart summarizes the communication between the different software components.

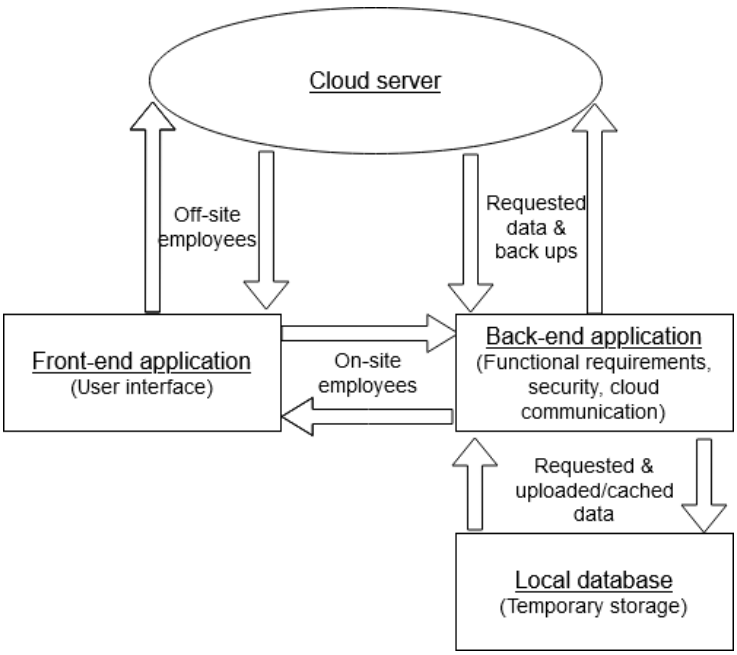

Figure 4. Communication between the software components

3.2 Security software design

Describe the software components and configuration supporting the security and privacy of the system.

In this section we will go over the different software component like previously, this time focusing on the security requirements that each component has, the security mechanisms that each component needs to fulfil said requirements and a detailed description of how a security mechanism would work along the application.

Again, just like before, the cloud service is not included in this as its security is guaranteed by the host. We will however mention communication between the application and the cloud service and the communication within the application itself.

- Local server and back-end application
  Security requirements: Identification and authorization, Auditing, Confidentiality, Data integrity, Access control and Non-repudiation.
  Security mechanisms: Digital certificates, Message digests and digital signatures, Cryptography, Digital Integrity, Authentication exchange, Routing control, Access control and Logging.

  - The local server should be capable of identifying users and authorizing their various identities when they're using the front-end application to communicate with it. In addition, it should be able to authenticate their identities and audit all their requests and operations. It should also maintain confidentiality and integrity of its data.
    Security mechanisms to achieve the requirements and further explanations:
    * **Digital certificates and signatures:** In these mechanisms, the local server checks the certificate of the user who is trying to communicate with it to establish his identity. These certificates are usually made by encrypting the digests that are made from the entire information using hash functions. These digests then encrypted using the users private key and the user then adds his identity with a signature right after the digest so that people know to use his public key when checking the certificates. There should be a certificate awarding authority who starts the loop of every employee certifying the identity of the employees below him. The highest entity could be either the president of the company or some other administrator with high privileges and responsibilities. These mechanisms establish the identity of the user and also contribute to authentication, auditing and non-repudiation (the inability of users to deny accountability).
    * **Authentication exchange and cryptography:** In these mechanisms, the local server exchanges cryptography keys using asymmetric encryption as the exchange tool. The local database will

use the public key of the person to be authenticated to encrypt a generated symmetric key. The local database will send this encrypted symmetric key to the owner of the public key, establishing an encrypted communication between the local database and the user and establishing authentication to a certain degree of trust. Further authentication is applied after a secure communication is established such as comparing credentials, IPs, MAC addresses and/or digital certificate checks. These mechanisms contribute mainly to authentication and to some degree to identification and non-repudiation.

* **Cryptography, Digital integrity and routing control:** In these mechanisms, the local server will try to maintain confidentiality and integrity of the data inside of it. The local server should use cryptographic algorithms to encrypt its most critical data including credentials, keys and data related to Loco news. The developers should have a cryptologist who guides them to safely use known algorithms and protocols. The local database should also use hash functions to create hashes and digests of its data, it would then use these hashes and digests for integrity checks. Those integrity checks should occur regularly and preferably after each operation of importance. Since the hash functions are efficient, doing theses checks won't affect performance. The local database should also handle routing control by enabling selection of particular physically secure routes for certain data and allows routing changes, even though the LAN is mostly detached from any outside networks except to the cloud through only the local server, this security mechanism is an extra layer of confidentiality and authentication during transport. This will especially be helpful when there's a suspicion of a security breach by someone inside the LAN. These mechanisms contribute mainly to confidentiality and integrity; however, they might play a minor role in fulfilling other security requirements such as auditing.
* **Access control and logging:** In these mechanisms, the local server will attempt to audit everything that happens within its boundaries including incoming and outgoing communications. The server should have several log files with backups ready in case of failure. These files should be protected with cryptography to ensure integrity and confidentiality. The access control security mechanism will ensure that the access control security requirement is met by using a privilege matrix or similar structures to enforce access control in the front and back end application. These methods contribute mainly to auditing and play some other roles in fulfilling security requirements, such as non-repudiation.

- Front-end application
  Security requirements: In addition to what's previously mentioned, Complete mediation, Privilege control, Access control, Isolation of components, Ease of use.
  Security mechanisms: In addition to what's previously mentioned, Reference monitor.

  - The front-end application should be capable of completing its share of the deal concerning identification and authentication, Auditing, Confidentiality, Data integrity and Access control. For the most part, its job is to provide the relative data to the back-end application in a secure manner. It should also have other qualities such as mediation, communication with other front-end applications and access and privilege managements. **Security mechanisms to achieve the requirements and further explanations:**
    * **Authentication and identification:** The front-end application should use known, tried and tested security packages and mechanisms such as The Kerberos and NTLM protocols that are used to solicit identity information from the user and to authenticate network logins and client/server connections. It would then send it through a secure connection such as the secure socket layer to the back-end application where it will be treated further.
    * **Complete mediation:** The front-end application should be able to check the privileges of a given user when a certain operation is demanded. It would use mechanisms such as reference monitor or any other mechanism that achieve a similar goal. The advised reference monitor could be established using different approaches such as:
      · Use an interpreter. The target program does not execute directly on the underlying hardware but instead is interpreted by another program. This is effectively the approach employed by the Java Virtual Machine. Although it allows a wide range of security policies to be implemented, it's not efficient and tends to be slow because of the added instructions to every operation.
      · Use a wrapper. A wrapper is an environment that intercepts (and interprets or redirects) only some of the instruction issued by the target program. hence why it's faster than an interpreter.
      · Exploit hardware to intercept relevant instructions [12].
    * **Privilege control and Access control:** The program should complete its part of the job when it comes to access control and privilege control by using the relative resources on the server to enforce the policies described earlier in this document. The reference monitor described earlier can also handle Access control and

privilege control on the front-end side.

* **Isolation of components and Ease of use:** The front-end application must logically isolate all of its previously described components to assure minimal risk and damage. It should also be as intuitive as possible to reduce the risk of compromising security policies. The developing company might find it interesting to employ the help of experts in human psychology to help with implementing the user interface to achieve the best results possible.

4 Use case scenario

This chapter should provide examples that show the operation of the system.

4.1 Use case: Login and fetch information.

This use case will happen most likely on a daily basis whenever it's a working day. It consists of an employee who wants to fetch some information like a name, a telephone number or even a whole article that needs to be printed. Before we discuss the scenario in detail, let us introduce the elements in play.

- **The actor:** An employee from Loco news.
- **Secondary actor:** The cloud-based database server.
- **The system:** The Loco news application.
- **Use cases:** Login, Fetch information.

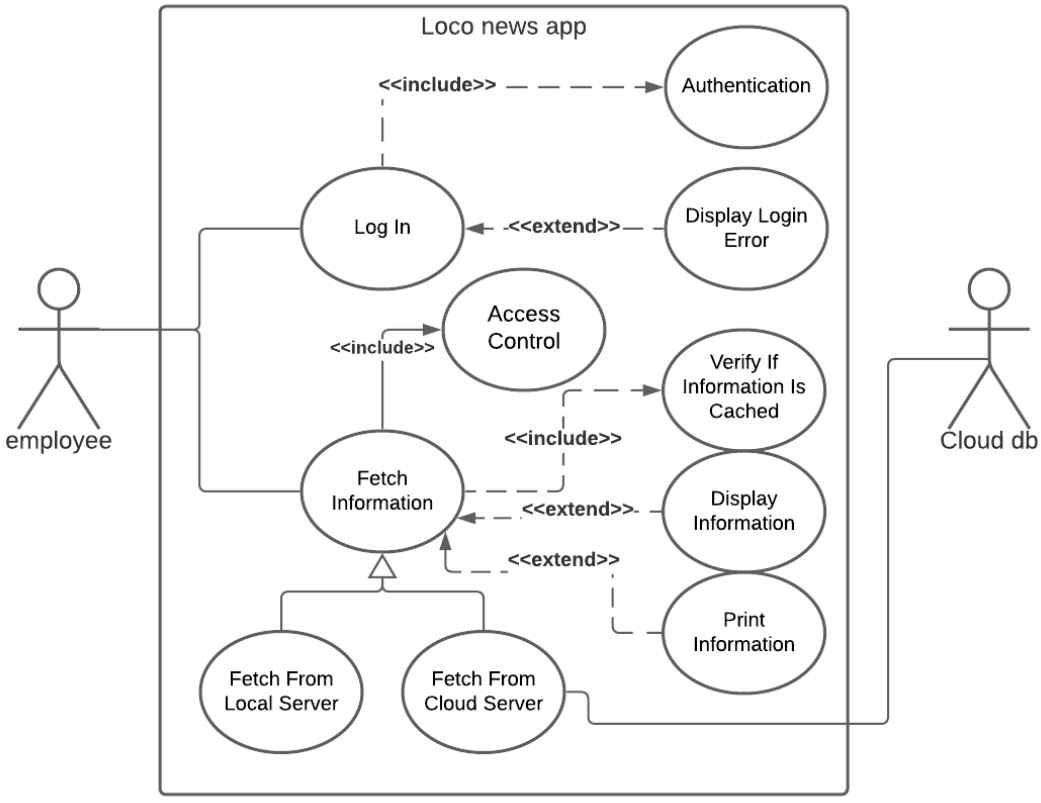

Figure 5. A UML diagram of the login and fetch information use case.

Following Figure 6 from left to write, Let's describe this scenario step by step.

- The actor, the employee, is asked to print an article that the boss wants. He sits on his desk, opens the application and starts the login process.
- The login window asks the employee for his credentials, which can be of three categories (know, have, is) described earlier in this document. The employee provides the required information and confirms. this starts the authentication process that is described in the security mechanisms section of this document. This authentication process is illustrated in the figure noting that it is a dependency of the Login use case, which means that it must be executed for the Login use case to be executed. From this point forward, every action happens is logged by the application including this one.
- In case of a failure of authentication, the application will display an error message as described by the figure.
- Once the authentication is complete, the employee will ask the program for the article in question. This action is illustrated by the Fetch Information use case in the figure.
- The application receives the command and the arguments demanded. It will recognize that it is indeed a fetch command with an article as an argument.
- The application checks that the employee has the correct privileges to demand such information as described as Access Control on the figure. If he does not have the rights, the application will display an error message informing him of the reality.
- In case the employee has the right access privilege, the application will proceed to check the local server, that is used as cache, for the demanded article. If the article.
- If the article is there, the application checks the employee's privileges one more time to avoid incomplete mediation. If the employee passes the check, the information is provided.
- If the article demanded is not cached in the server, the application will prepare to call on the cloud server using the methods and the authentication procedures that the host specified.
- The application receives the article from the host server and provides it to the employee given the employee passes the second access control security check.
- After the article is delivered to the employee and printed, the application will continue to log any action until the user has logged out.
- As soon as the employee logs out, our use case is over.

4.2 Use case: Uploading and downloading data

This use case will likely happen multiple times on every working day. Employees write articles that need to be uploaded, or they have to download certain data. There are two possible scenarios:

- **The employee is on-site**
  In this scenario, the front-end application on the employee's computer will communicate with the back-end application on the server on the local network. In case the employee uploads data, the back-end application will then store the data in the local database and make a back up of the database to the cloud on regular intervals. In case the employee requests data, the back-end application will check whether that data is in the local database. If so, it will fetch it from the database and send it back to the employee. If not, it will fetch the data from the cloud and send it back to the employee. (Assuming the employee is authorized to make these requests.)

- **The employee is off-site**
  In this scenario, the front-end application on the employee's computer will communicate directly with the cloud. Data to be uploaded or downloaded will be fetched or send to directly to the cloud from the employee's computer and current network.

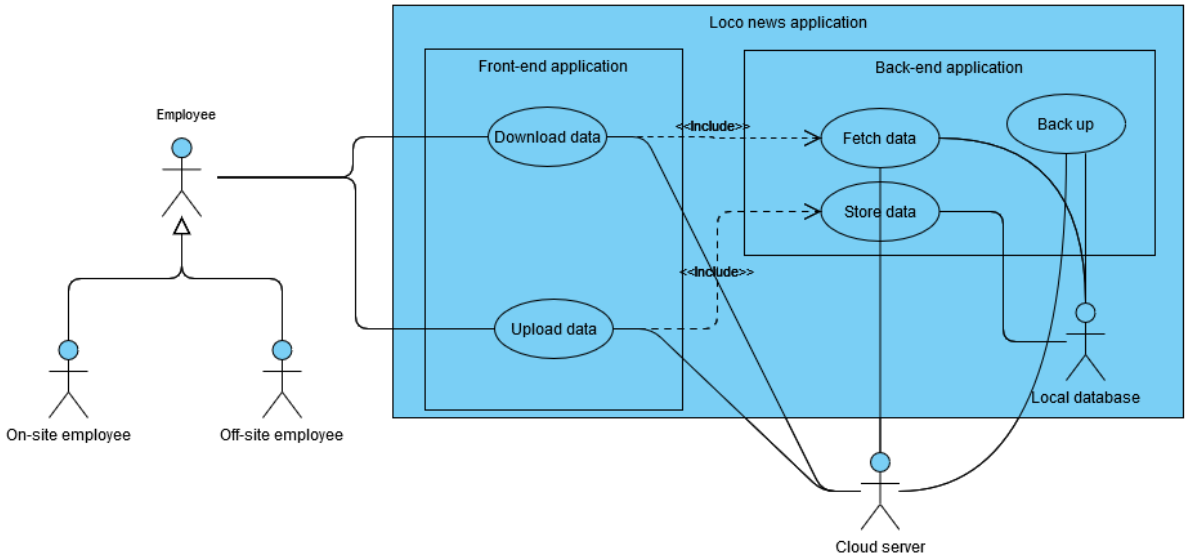

Figure 6. Employee uploading or downloading data

Let's walk through this diagram step by step.

  - **The employee requests to download data**
    - **The employee is off-site**
      The data is fetched from the cloud and send back to the employee.

    - **The employee is on-site**
      The request is sent to the back-end application, which will fetch the requested data from the local database or the cloud server.

- **The employee uploads data**
    - **The employee is off-site**
      The data is sent to the cloud.
    - **The employee is on-site**
      The data is sent to the back-end application, which stores it in the local database. The local database is periodically backed-up to the cloud.

5 Practices that endanger security

The following section lists some of the organization's practices that endanger information security, as well as advice on how to mitigate the risk.

- **Practice:** The server room is located in the basement. This entices flooding danger.
  Advice: Move the server room to a secure location, or protect against flooding risks.

- **Practice:** The servers are close to archives and old furniture, this material is flammable.
  Advice: Protect the servers from these materials with a fire-proof wall.

- **Practice:** Every employee has full admin privileges on their computer. This could lead to employees abusing their computer (e.g. installing malware), either intentional or unintentional.
  Advice: Regulate the privileges on employees' computer, with an administrator overseeing all privileges.

- **Practice:** The servers have no protection against physical access except for a normal lock.
  Advice: Strong locks and checks should be implemented before physical access to the server room is granted. Physical access privileges should be granted only to administrators.

- **Practice:** Employees bring their own personal devices and connect them to the company network (e.g., phones) or to their work computers (e.g., USB drives). This might lead to malware transferring from the employees' personal devices to the company properties.
  Advice: Create a separate network for the employees' personal devices, separated from the work computers. Furthermore, give the employees training in information security.

- **Practice:** The router uses WPA-PSK which might be insecure [13] [14]. This could allow unauthorized access to the network or network abuse.
  Advice: Acquire a new router with a modern security protocol.

- **Practice:** The server runs on out-dated software and operating systems. This could entice security vulnerabilities (Common Vulnerabilities and Exposures (CVEs)).
  Advice: Update the servers to new operating systems and patch them regularly.

- **Practice:** The server stores (potentially sensitive) data non-encrypted. This can lead to the data being compromised. Furthermore, there is no current back-up policy of the database. **Advice:** These problems are transitory, as they will all be solved when the new system is implemented. However, in the meantime, these problems should be addressed by encrypting sensitive

data, and creating a back-up on separate hardware that is stored offline on another location.

- **Practice:** Employees' work laptops are brought outside. They can be stolen and data can be compromised.
  Advice: Secure and encrypt all data on employees' computers thoroughly, for example with password protection.

References

[1] P. Pickett, "What is sql?" Jul 2020. [Online]. Available: https://www.thebalancecareers.com/what-is-sql-and-uses-2071909

[2] B. Dobran, "Linux vs. microsoft windows servers, the ultimate showdown," Oct 2020. [Online]. Available: https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers

[3] "Introduction to google drive api — google developers." [Online]. Available: https://developers.google.com/drive/api/v3/about-sdk

[4] "Php mysql database." [Online]. Available: https://www.w3schools.com/php/php_mysql_intro.asp

[5] W. i. J. B. M. Tyson and M. Tyson, "What is jdbc? introduction to java database connectivity," Apr 2019. [Online]. Available: https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-connectivity.html

[6] T. P. Group, "What is php?" [Online]. Available: https://www.php.net/manual/en/intro-whatis.php

[7] opensource.com, "What is java?" [Online]. Available: https://opensource.com/resources/java

[8] J. Reock, "Considering openjdk? how to decide," Nov 2019. [Online]. Available: https://www.openlogic.com/blog/considering-openjdk-how-decide

[9] S. Soni, "How to run a php file," Apr 2020. [Online]. Available: https://code.tutsplus.com/tutorials/how-to-run-a-php-file--cms-34769

[10] "Java swing tutorial: How to create a gui in java with examples." [Online]. Available: https://www.guru99.com/java-swing-gui.html

[11] "10 best javascript frameworks to use in 2020." [Online]. Available: https://hackr.io/blog/best-javascript-frameworks

[12] P. F. B. Schneider, "Implementing complete mediation." [Online]. Available: https://www.cs.cornell.edu/courses/cs513/2000SP/NL05.html

[13] News, L. A. T. AuthorNews, and L. A. T. Author, "Wpa2-psk is not good enough," Sep 2020. [Online]. Available: https://www.securew2.com/blog/wpa2-psk-is-not-enough/

[14] "Wpa vs wpa2." [Online]. Available: https://www.diffen.com/difference/WPA_vs_WPA2